Aalto University
School of Chemical
Technology

School of Chemical Technology
Degree Programme of Chemical Technology

Lauri Pöri

COMPARISON OF TWO INTERIOR POINT SOLVERS IN MODEL PREDICTIVE CONTROL OPTIMIZATION

Master's thesis for the degree of Master of Science in Technology submitted for inspection, Espoo, 21 November, 2016.

Supervisor **Professor Sirkka-Liisa Jämsä-Jounela**

Instructor **M.Sc. Samuli Bergman**
M.Sc. Stefan Tötterman

Aalto University
School of Chemical
Technology

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author	Lauri Pöri
Title of thesis	Comparison of two interior point solvers in model predictive control optimization
Department	Department of Biotechnology and Chemical Technology
Professorship	Process Control
Code of professorship	KE-90
Thesis supervisor	Professor Sirkka-Liisa Jämsä-Jounela
Thesis advisor(s) / Thesis examiner(s)	M.Sc. Samuli Bergman, M.Sc. Stefan Tötterman
Date	21.11.2016
Number of pages	85
Language	English

Abstract

Model Predictive Control (MPC) uses a mathematical programming problem to calculate optimal control moves through optimizing a cost function. In order to use MPC in a real industrial environment, the optimization solver needs to be efficient for keeping up with the time restrictions set by the control cycle of the process and robust for minimizing the inoperability of the controller in all situations.

In this thesis, a comparison of two interior point solvers in MPC optimization is conducted. The studied solvers are IPOPT optimization software package versions 1.6 and 3.12. The optimization software is tested using the NAPCON Controller which is an advanced predictive controller solution that utilizes the receding horizon model predictive control strategy.

The literature review of this thesis focuses on the basics of nonlinear optimization and presents the fundamentals of the Newton's method, Sequential Quadratic Programming (SQP) methods and Interior Point (IP) methods. State-of-the-art nonlinear programming software is reviewed and two commercially available nonlinear SQP solvers and two nonlinear IP solvers are presented. Additionally, the basics of model predictive control are covered.

The experimental section describes the used software and the implementation of the tests as well as the results. The solvers are tested using a hydrogen treatment process model that is simulated. The practical testing consists of cold start and warm start tests in which the time and number of iterations used for the optimization is tracked. The cold start tests constitute as the worst case scenario test in which the controller starts in a very unfavourable state which violates many constraints and the initial guess for the optimization problem is bad. The test is also run with different amounts of variables to also see how the problem size affects the optimization. In the warm start test, the NAPCON Controller is used to control the test process and a sequence of step changes. In addition, limit changes are executed to test how the solvers perform in real time MPC optimization. Warm start optimization utilizes the solutions from the previous iteration to solve the current optimization problem faster. The test is conducted to see how normal control actions and difficult circumstances affect the performance in continuous optimization.

The tests show that the IPOPT v. 3.12 outperforms the IPOPT v. 1.6 in each test by a large margin. The cold start tests show that the performance difference increases as the number of variables in the optimization problem increase and the growth in CPU time is approximately linear. The warm start tests show that the IPOPT v. 3.12 is also faster in continuous MPC optimization. When introducing disturbances IPOPT v. 3.12 solves the problem faster and returns to normal operation states more quickly than IPOPT v. 1.6.

Keywords MPC, optimization, interior point method, IPOPT, NAPCON Controller

Aalto-yliopisto
Kemian tekniikan korkeakoulu

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi

Diplomityön tiivistelmä

Tekijä Lauri Pöri		
Työn nimi Kahden sisäpisteratkaisijan vertailu mallinprediktiivisen säädön optimoinnissa		
Laitos Biotekniikan ja kemian tekniikan laitos		
Professuuri Prosessien ohjaus		**Professuurikoodi** KE-90
Työn valvoja Professori Sirkka-Liisa Jämsä-Jounela		
Työn ohjaaja(t)/Työn tarkastaja(t) DI Samuli Bergman, DI Stefan Tötterman		
Päivämäärä 21.11.2016	**Sivumäärä** 85	**Kieli** englanti

Tiivistelmä

Mallinprediktiivisessä säädössä (Model Predictive Control, MPC) käytetään matemaattista ohjelmointiongelmaa laskemaan optimaaliset ohjausliikkeet optimoimalla kustannusfunktiota. Jotta MPC:tä voidaan käyttää oikeassa teollisuusympäristössä, optimointiratkaisijan täytyy olla tehokas pystyäkseen kunnioittamaan prosessin asettamia aikarajoitteita säätösykleille sekä luotettava minimoidakseen säätimen toimimattomuuden kaikissa tilanteissa.

Tässä työssä verrataan kahta sisäpisteratkaisijaa MPC-optimoinnissa. Tutkitut ratkaisijat ovat IPOPT-optimointiohjelmiston versiot 1.6 ja 3.12. Optimointiohjelmistot testataan NAPCON Controller -säätöohjelmistolla, joka hyödyntää väistyvän horisontin (receding horizon) säätöstrategiaa.

Kirjallisuuskatsauksessa keskitytään epälineaarisen optimoinnin perusteisiin ja esitetään keskeiset asiat Newtonin menetelmästä, sekventaalisesta quadraattisesta ohjelmoinnista (Sequential Quadratic Programming) ja sisäpistemenetelmästä (Interior Point method). Työssä esitellään tunnetuimpia kaupallisesti saatavia epälineaarisia ratkaisijoita, joista kaksi hyödyntää sekventaalista quadraattista ohjelmointia ja kaksi sisäpistemenetelmää. Myös MPC:n perusteet esitetään.

Kokeellisessa osassa kuvataan käytetyt ohjelmat, kokeiden toteutus sekä kokeiden tulokset. Ratkaisijoita testataan simuloidussa vetykäsittelyprosessissa. Kokeet koostuvat kylmä- ja kuumakäynnistystesteistä, joissa optimointiin käytettyä aikaa ja iteraatioiden määrää mitataan. Kylmäkäynnistystestissä säädin aloittaa huonosta alkutilasta, jossa monia rajoitteita rikotaan ja optimointiongelman alkuarvaus on huono. Tämä testi ajetaan eri muuttujamäärillä, jotta voidaan tutkia ongelman koon vaikutusta optimointiin. Kuumakäynnistystestissä NAPCON Controller -säädintä käytetään prosessin säätöön. Prosessiin ajetaan sekvenssi askelmuutoksia, joilla testataan, miten ratkaisijat suoriutuvat reaaliaikaisesta MPC-optimoinnista. Tässä testissä tutkitaan, miten normaalit säätötoimenpiteet ja vaikeat prosessitilat vaikuttavat suorituskykyyn jatkuvassa optimoinnissa.

Kokeista käy ilmi, että IPOPT v. 3.12 suoriutuu paljon paremmin kaikista testeistä verrattuna IPOPT v. 1.6:een. Kylmäkäynnistystesteissä nähdään, että suorituskykyero kasvaa muuttujien määrän lisääntyessä, ja kasvu suoritusajassa on lähes lineaarista. Kuumakäynnistystesteissä näkyy, että IPOPT v. 3.12 on myös nopeampi jatkuvassa MPC-optimoinnissa. Kun häiriöitä tuotetaan, IPOPT v. 3.12 ratkaisee optimointiongelmat tehokkaammin ja palaa normaalin tilaan nopeammin kuin IPOPT v. 1.6.

Avainsanat MPC, optimointi, sisäpistemenetelmä, IPOPT, NAPCON Controller

Preface

This master's thesis was authored at the Technology and Product Development of Neste Jacobs Oy in Porvoo during the time between January 2015 and October 2016. This thesis was supervised by professor Sirkka-Liisa Jämsä-Jounela from Aalto University.

First of all, I want to thank my instructors M.Sc. Samuli Bergman and M.Sc. Stefan Tötterman for all the guidance and feedback they gave me throughout the whole process. I also want to express my gratitude to professor Sirkka-Liisa Jämsä-Jounela for all the guidance and feedback she has given me during this thesis as well as during my studies at the Aalto University. In addition, I want to extend my gratitude to D.Sc. Alexey Zakharov from Aalto University, who provided me important feedback about the layout and content of the thesis, and to Senior Associate Martti Ojala from Neste Jacobs Oy, who gave me valuable help with the programming tasks involved in the thesis.

Special thanks to all my colleagues for making Neste Jacobs such an enjoyable place to work and finally, I also want to thank my family and friends for all the support they have given me.

Enjoy,

Lauri Pöri

Table of Contents

Literature Part

1 Introduction

Model Predictive Control (MPC) has been used in petrochemical industry since 1970s [1]. The complicated processes in the industry with large time delays, strong cross-interactions, nonlinearities, and numerous constraints have promoted the use of more sophisticated control strategies than the classical PID control. The key advantage in MPC is the ability to use multiple inputs and outputs to predict the future process states while taking in account the process constraints. At the heart of MPC is the calculation of control moves as a mathematical programming problem that optimizes a cost function of set objectives over a prediction horizon while respecting the given constraints. The cost function may include multiple objectives such as minimizing error between targets and predictions, reducing control input deviations and even maximizing economic objectives.

According to Nagy and Allgöwer [2] "strong trends such as demands for improved product quality, stricter environmental regulation, development of new complex processes, and plants, which are highly integrated to improve energy efficiency, make the control of processes increasingly difficult". The technological advancements in processing, transferring and storing data have made it possible to achieve company wide data integration through all levels of automation hierarchy which enables the use of plant-wide optimization strategies to maximize profits even on an enterprise level. To tackle such problems and to take advantage of the large-scale data integration the use of larger and more complex models in control strategies is required. This translates into more demanding optimization problems in MPC, and as a result, more efficient and robust optimization algorithms for large-scale optimization are needed to keep up with the time restrictions set by the online optimization of the MPC problem and to uphold the reliability of the solution even with large and complex optimization tasks.

Neste Jacobs Oy offers an Advanced Process Control (APC) solution called NAPCON Controller, which implements an advanced MPC software. The aim of this thesis is to apply a modern, updated version of the state-of-the-art interior point optimization package IPOPT for solving the MPC optimization problem of the NAPCON Controller in real-time. The tests are carried out using a hydrogen treatment unit process model. The reference used for evaluating the performance of the updated IPOPT package is the earlier adaptation of the same package coupled with the NAPCON Controller. The focus of the testing is in the efficiency and robustness of the optimization software.

The structure of the thesis consists of a literature part and an experimental part. The literature part explains some fundamental theory for nonlinear optimization (Chapter 2) which is then followed by an introduction and explanation of a few well-known optimization strategies (Chapter 3). Lastly, the basics of model predictive control are covered in Chapter 4.

The experimental part first introduces the objectives of the experiments (Chapter 5) and then describes the used software (Chapter 6). The implementation of the optimization software is explained in Chapter 7. The testing and evaluation processes are described in Chapter 8 and the results of the tests are presented in Chapter 9. The summary and conclusions are presented in Chapter 10 and finally, propositions for further study are suggested in Chapter 11.

2 Nonlinear Programming

Nonlinear Programming (NLP) is a process of optimizing a continuous system consisting of equality and/or inequality constraints by minimizing or maximizing an objective function.

The general NLP problem is of the following form:

$$\min f(x) \quad \textbf{(2.1a)}$$

s.t.

$$h(x) = 0, \quad \textbf{(2.1b)}$$
$$g(x) \leq 0, \quad \textbf{(2.1c)}$$

where $f : R^n \rightarrow R$ and $x \in R^n$. Similarly, the NLP can be expressed as a maximization problem with inequality constraints $g(x) \geq 0$. [3 p. 2].

Minimization problem can be transformed into a maximization problem by utilizing the ordering property of real numbers. Since f is maximized at x if $f(x) \geq f(y)$ for all y $(x, y \in R)$, multiplying the condition by -1 leads to $-f(x) \leq -f(y)$, which equals to $\min -f(x)$. Therefore $\max f(x) = \min -f(x)$.

Functions $f(x)$, $h(x)$ and $g(x)$ are called objective function, equality constraint function and inequality constraint function, respectively. In the scope of this thesis, these functions are assumed to be twice differentiable so that powerful derivative utilizing methods and optimality conditions can be applied. If the assumption cannot be made, then derivative-free optimization (DFO) methods, such as direct search methods, pattern search methods or phenomenological methods, are required [4].

Optimization problems can be divided to different categories according to the nature of their functions and constraints as shown in Figure 2-1. The differentiable NLP problems can be further divided into convex and non-convex problems. More details about convexity will be provided in Chapter 2.1. Convex problems are special in a way that any found local optimum is also the global optimum. The two special cases of convex programming problems are Linear Programming (LP) problems and Quadratic

Programming (QP) problems, which both have their own specialized algorithms. A LP problem has purely linear components as opposed to NLP problem, which has at least one nonlinear component. A QP problem is similar to LP problem but with the addition of a quadratic term in the objective function. Convex cases of these problems can be solved in a finite number of steps. [4].

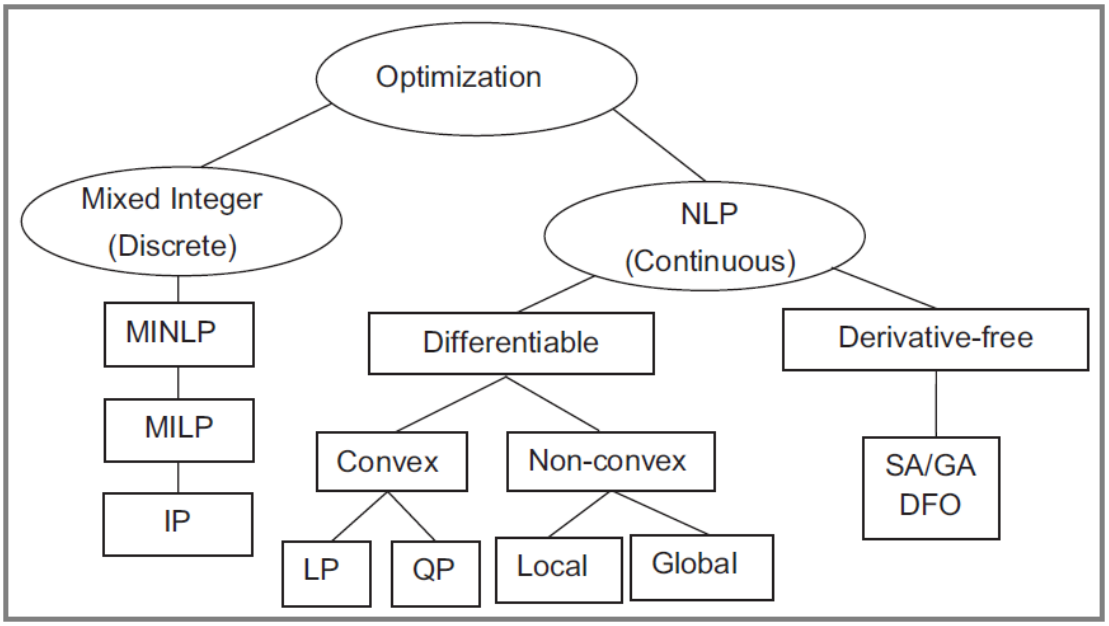

Figure 2-1. Classes of optimization problems. [4].

Correspondingly, optimization problems containing discrete decision variables, in addition to continuous ones, are called Mixed Integer (MI) problems and they can be divided into three subcategories. These are, in order, Mixed Integer Nonlinear Programming (MINLP) problems, Mixed Integer Linear Programming (MILP) problems and Integer Programming (IP) problems. MINLP problems have at least one nonlinear function whereas MILP problems consist of purely linear functions and the fully discrete IP problems contain only integer functions. [4].

The literature part of this thesis, however, focuses on methods to solve NLP problems; more specifically the focus is on methods that can be utilized in solving the optimization problem in MPC applications.

2.1 Convex functions

Function $\theta(x)$ of x in some domain X, is considered convex if, and only if, all points $x_1, x_2 \in X$ [5 p. 4]:

$$\theta(ax_1 + (1-a)x_2) \leq a\theta(x_1) + (1-a)\theta(x_2),\ a \in (0,1). \quad \textbf{(2.2)}$$

In other words, a function is convex if, for each point on the graph of the function, a line can be drawn to any other point on the graph without crossing over the graph at any point. In more mathematical terms, the epigraph of a convex function must be a convex set. An example of a convex and a non-convex set is presented in Figure 2-2.

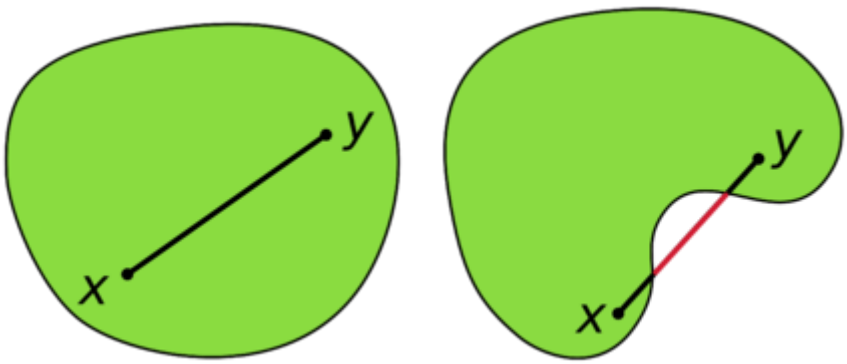

Figure 2-2. A convex set and a non-convex set. The convex set is presented on the left and the non-convex set on the right. [6], [7].

The NLP problem presented in (2.1) can be considered convex if it has a convex objective function $f(x)$ and a convex feasibility region. For a feasibility region to be convex the inequality constraint function $g(x)$ is required to be convex and the equality constraint function $h(x)$ linear. [5 p. 4].

The primary benefit of convexity is that, for any convex problem, a local solution is also necessarily a global solution. For non-convex problems this is not true and multiple local solutions may exist, and therefore more sophisticated and computationally much heavier methods are required to find the global solution. [5 p. 4].

2.2 Linear Independence Constraint Qualification

The Linear Independence Constraint Qualification (LICQ) is used to ensure that when linearizing nonlinear constraints the limiting directions of active constraints are reliable at the solution(s). The linear independence of the constraint gradients guarantees that the constraint set remains well-defined so that Karush-Kuhn-Tucker conditions are solvable and have a unique solution. The LICQ is defined by linear independence of the constraint gradients $\nabla g_i(x^*), \nabla h_i(x^*), i \in A(x^*)$ at a trial solution x^*, where $A(x^*)$ is the set of indices of active constraints. [5 p. 77]. In other words, the matrices $\nabla g(x^*), \nabla h(x^*)$ (of active constraints) have full rank at the solution x^*.

2.3 Karush-Kuhn-Tucker Conditions

The Karush-Kuhn-Tucker (KKT) optimality conditions (also known as first-order necessary conditions) present the circumstances that need to uphold for a solution of a constrained nonlinear programming problem. These conditions were first published by Kuhn and Tucker (1951) in [8]. Later, it was discovered that William Karush had arrived to same conclusions in his unpublished master's thesis already in 1939. Therefore the

conditions are now called the Karush-Kuhn-Tucker conditions. Kuhn analyzes the origins of their discoveries in a historical view published in 1976. [9].

For the NLP presented in (2.1), the Karush-Kuhn-Tucker conditions at a solution (x^*, u^*, v^*) can be defined as follows:

$$\nabla_x L(x^*, u^*, v^*) = \nabla f(x^*) + \sum_{i=1}^{m} u_i^* \nabla g_i(x^*) + \sum_{i=1}^{p} v_i^* \nabla h_i(x^*) = 0\,, \qquad \textbf{(2.3a)}$$

$$h_i(x^*) = 0, \qquad i = 1, \dots, p, \qquad \textbf{(2.3b)}$$

$$g_i(x^*) \leq 0, \qquad i = 1, \dots, m, \qquad \textbf{(2.3c)}$$

$$u_i^* \geq 0, \qquad i = 1, \dots, m, \qquad \textbf{(2.3d)}$$

$$u_i^* g_i(x^*) = 0, \qquad i = 1, \dots, m, \qquad \textbf{(2.3e)}$$

where $L(x^*, u^*, v^*)$ is the Lagrangian function of (2.1) and (u^*, v^*) are the so-called KKT multipliers. If the LICQ holds at the solution the KKT multipliers are unique. The condition (2.3e) is called the complementary condition which signifies whether inequality constraints are active or not. In the latter case the KKT multipliers are zero and thus do not affect the gradient of the Lagrangian function. As a result, the inactive constraints are irrelevant to the solution and can be discarded from the optimization problem to simplify it. [5 pp. 70-71].

For a equality constrained problem that doesn't have any inequalities, the KKT conditions are defined as in (2.3) except without any terms containing g_i or u_i^*. These conditions are then called the Lagrange optimality conditions (or sometimes primal optimality conditions if a primal-dual system is considered) to denote the absence of inequality constraints. [5 p. 151].

2.4 Second-order Conditions

In nonlinear programming the information from first derivatives is not always enough to distinguish the difference between saddle points and local optima. This promotes the use of information stored in the second derivatives of the objective and constraint

functions to advance (or conclude) the optimization process. The utilization of second-order conditions requires that the functions are at least twice differentiable. Essentially, the second-order conditions exploit the curvature of the Lagrangian function to determine if moving to a feasible direction w will increase or decrease the objective function. Consequently, for any point that is considered as a solution for (2.1), and for which the KKT conditions and LICQ holds, the necessary second-order conditions can be defined as

$$w^T \nabla_x^2 L(x^*, u^*, v^*) w \geq 0, \qquad for\ all\ w \neq 0, \tag{2.4a}$$
$$\nabla h(x^*)^T w = 0, \tag{2.4b}$$
$$\nabla g_i(x^*)^T w = 0, \qquad i \in \{i | g_i(x^*) = 0, u_i^* > 0\}, \tag{2.4c}$$
$$\nabla g_i(x^*)^T w \leq 0, \qquad i \in \{i | g_i(x^*) = 0, u_i^* = 0\}. \tag{2.4d}$$

The necessary second-order conditions are required to hold at all solutions. On the other hand, the sufficient second-order conditions, that require the inequality in (2.4a) to be strict (but do not require LICQ), are not mandatory for the solution. [10 pp. 330-333]. However, if a point (x^*, u^*, v^*) satisfies the sufficient second-order conditions, then it is guaranteed to be a solution to the optimization problem [10 p. 305].

3 Optimization Methods

The solving of nonlinear optimization problems naturally requires proper algorithms to deal with the difficulties rising from nonlinearities. The algorithms presented in this chapter are general examples of these optimization method categories and try to offer a basic understanding of how these methods generally work. Numerous enhancements are available to these methods to tackle many problems that might arise in nonlinear optimization such as global convergence, matrix singularity and definiteness, infeasible trial solutions and availability of second derivatives. More information on these issues can be found in literature such as [5] and [10].

The first algorithm presented is the Newton's method, which is the basis of many optimization algorithms including both Sequential Quadratic Programming (SQP) and Interior Point (IP) methods presented in this chapter. The simple and effective method has a good theoretical background and is proven in practice with some modifications. The fast, quadratic convergence rate is a key property that is generally inherited by the algorithms utilizing the method.

Secondly, the SQP method is presented, which utilizes a series of quadratic approximations of the original nonlinear problem to calculate steps that will converge to a solution. The method uses the so-called active set strategy to deal with the complementary KKT conditions. The simplicity and applicability of SQP makes it one of the most popular NLP algorithms.

Finally, the IP method is presented. This method utilizes a so-called barrier function to keep the solution strictly in the feasible region and thereby avoiding the decisions about the active constraint sets. While this means that all constraints are considered at each iteration, the constant nonzero structure of the matrices allows a straight-forward exploitation of the matrix structures in terms of, for example, sparsity. The interior point method deals with the complementary conditions through relaxation of the constraints by applying the barrier function.

3.1 Newton's Method

Newton's method is an iterative algorithm for finding local optima of an unconstrained NLP by using its first and second derivatives. It provides fast local convergence and many concepts in Newton-type methods can be extended to fast constrained optimization [5 p. 34], making it an important touchstone method in the optimization literature.

Let the unconstrained optimization problem be $\min f(x)$. The algorithm takes advantage of the Taylor series expansion by making a quadratic approximation of the objective function (assuming it's twice differentiable):

$$f(x^k + p) \approx f(x^k) + p^T \nabla f(x^k) + \frac{1}{2} p^T \nabla^2 f(x^k) p, \quad \textbf{(3.1)}$$

where p is a scalar vector, $\nabla f(x^k)$ is the gradient of $f(x^k)$ and $\nabla^2 f(x^k)$ is the Hessian matrix of $f(x^k)$. The search direction p is found by minimizing (3.1)

$$\min_p Q(p) = f(x^k) + p^T \nabla f(x^k) + \frac{1}{2} p^T \nabla^2 f(x^k) p, \quad \textbf{(3.2)}$$

where $f(x^k)$ is a fixed scalar, $\nabla f(x^k)$ is a fixed vector and $\nabla^2 f(x^k)$ is a fixed symmetric matrix. Assuming that $\nabla^2 f(x^k)$ is positive definite, a solution to problem (3.2) is found by solving the following system of linear equations (for p):

$$\nabla^2 f(x^k)\, p = -\nabla f(x^k). \quad \textbf{(3.3)}$$

The search direction p is, in this instance, the so-called Newton step or Newton direction. This method of calculating p defines the Newton's method algorithm for optimization. [11 p. 125].

On the other hand, if the Hessian $\nabla^2 f(x^k)$ is not positive definite, the inverse of the Hessian might not exist or it has negative eigenvalues. In the first case, the whole solution might not exist and in the second case, p cannot be guaranteed to be the

descent direction. Therefore, if the Hessian is not positive definite, a new modified Hessian has to be calculated that maintains the positive definiteness. Popular methods for modifying the Hessian include Modified Cholesky Factorization and Levenberg-Marquardt Correction. [5 pp. 39-42].

The above presented method is suited only for unconstrained optimization while constraints are a key part of many optimization problems. Therefore, an extension of the method is presented to include equality constraints. Equality constrained problems can be solved with Newton's method by utilizing the Lagrange optimality conditions.

Let the equality constrained NLP be defined as

$$\min f(x) \tag{3.4a}$$

s.t.

$$h(x) = 0 \tag{3.4b}$$

and the Lagrangian function as $L(x, v) = f(x) + v^T h(x)$. Then the Lagrange optimality conditions for (3.4) can be written as

$$\nabla_x L(x^*, v^*) = \nabla f(x^*) + \nabla h(x^*)^T v^* = 0, \tag{3.5a}$$
$$\nabla_v L(x^*, v^*) = h(x^*) = 0. \tag{3.5b}$$

The linearization of the nonlinear problem necessitates that LICQ must be satisfied for (3.5) to be the necessary conditions for a local optimum (x^*, v^*) to the original problem (3.4). This guarantees that the KKT multipliers (v^*) are unique at the optimal solution. Additionally, in order to assure a local optimum, the solution (x^*, v^*) needs to satisfy the necessary second-order conditions

$$d^T \nabla_x^2 L(x^*, v^*) d \geq 0, \qquad for\ all\ d \neq 0, \tag{3.6a}$$
$$\nabla h(x^*)^T d = 0. \tag{3.6b}$$

For the solution to be a strict local optimizer, the sufficient second-order conditions require the inequality in (3.6a) to be strict. [5 p. 92].

Finally, the next trial solution of the NLP can be obtained using Newton steps generated from solving the linear system (Taylor series expansion of (3.5))

$$\begin{bmatrix} \nabla_x^2 L(x^k, v^k) & \nabla h(x^k) \\ \nabla h(x^k)^T & 0 \end{bmatrix} \begin{bmatrix} p_x^k \\ p_v^k \end{bmatrix} = - \begin{bmatrix} \nabla_x L(x^k, v^k) \\ h(x^k) \end{bmatrix} \quad \textbf{(3.7)}$$

for $p^k = [p_x^k \; p_v^k]^T$, where the first matrix is the so-called KKT matrix. The iterates can then be updated as $(x^{k+1}, v^{k+1}) = (x^k + p_x^k, v^k + p_v^k)$.

Following is a general strategy that can be used to find a local optimum for a nonlinear function. Additional details that consider only the constrained case are in brackets ( ) or otherwise noted. The strategy is described here [11 p. 124], [5 p. 93]:

Choose an initial value x^0 (and v^0).

For each iterate x^k (and v^k), $k \geq 0$:

1. Iteration end tests:
    a. the norm of the gradient $\left|\left|\nabla f(x^k)\right|\right|$ (or $\left|\left|\nabla L(x^k, v^k)\right|\right|$) is below a chosen threshold
    b. the perturbation(s) $\left|\left|x^k - x^{k-1}\right|\right|$ (and $\left|\left|v^k - v^{k-1}\right|\right|$) is (are) below a chosen threshold
    c. Constrained problem only: KKT matrix is singular (failure)

    Note that usually multiple tests are required to pass simultaneously to stop the iteration.

2. Calculate a search direction
    a. Calculate a nonzero vector p^k from (3.3) (or (3.7))
        i. If p^k is zero and optimality conditions are fulfilled, stop iteration
3. Update the estimate
    a. Set $x^{k+1} = x^k + p^k$ (and $v^{k+1} = v^k + p_v^k$, note that $p^k = p_x^k$)
    b. Increase the iteration counter k

c. Go back to step 1

Newton's method can be further enhanced with better convergence properties from poor starting points by using line search or trust region methods. Line search methods try to iteratively find the largest step size (to update the estimate) that decreases the objective function sufficiently by trying smaller and smaller step size values [5 p. 47]. Similarly, the trust region methods adjust the iteration step length, but also modify the search direction to be dependent on the step length. Consequently, the trust region methods offer better convergence properties than line search methods but at the cost of computational efficiency. [5 pp. 52-53]. Further details and theoretical proofs of the functionality of these methods can be found in numerous publications, including [5] and [10].

3.2 Sequential Quadratic Programming Methods

Sequential Quadratic Programming (SQP) methods optimize NLPs by generating iteration steps from quadratic subproblems that utilize approximations of the original NLP and presumptions of active bounds. SQP methods are among the most popular methods for nonlinearly constrained optimization since they inherit the fast convergence properties of Newton-type methods while being applicable to both small and large scale problems. [5 p. 135].

Applying the quadratic Taylor series approximation to the objective function of the general NLP problem (2.1) and linear approximation to the constraints, yields the general QP subproblem

$$\min_{p} f(x^k) + \nabla f(x^k)p + \frac{1}{2}p^T\nabla^2 f(x^k)p \qquad \textbf{(3.8a)}$$

s.t.

$$\nabla h\left(x^k\right)^T p + h\left(x^k\right) = 0, \qquad \textbf{(3.8b)}$$

$$\nabla g\left(x^k\right)^T p + g\left(x^k\right) \leq 0. \qquad \textbf{(3.8c)}$$

Active set SQP methods can be categorized into two different categories: Inequality Quadratic Programming (IQP) methods and Equality Quadratic Programming (EQP) methods. These method-types differ by how the set of active constraints, or shortly active set, is determined and utilized. IQP methods solve the general QP subproblem (3.8) and determine the active set simultaneously during each iteration. The solution obtained at the previous iteration can be used as the initial guess to "warm start" the optimization of the QP subproblem related to the current iteration. If the guess is good, the warm starting strategy can decrease the computational costs of the subproblem significantly and therefore warm starting is always used in practice when possible. [10 pp. 530, 534].

EQP methods, on the other hand, decouple these calculations. The optimal active set is first estimated by solving an auxiliary subproblem or using rules based on Lagrange multipliers. The estimated active set is then used to solve an equality-constrained QP, where the active inequality constraints are imposed as equalities, to obtain the iteration step. The advantage of EQP methods is that the equality-constrained subproblem is computationally cheaper to solve than the general QP problem (3.8), especially when considering large-scale problems. [10 pp. 530, 534].

The method formulated in this chapter constitutes as an EQP approach. A bound constrained formulation of the NLP problem is used as it simplifies the determination of the active set. The method doesn't include any advanced features to guarantee global convergence or deal with infeasible starting points but it should describe the basic features of a SQP algorithm in sufficient detail.

The nonlinear programming problem introduced in (2.1) can be reformulated to an equivalent bound constrained problem [5 p. 134]:

$$\min f(x) \quad \textbf{(3.9a)}$$

s.t.

$$c(x) = 0, \quad \textbf{(3.9b)}$$

$$x_L \leq x \leq x_U, \quad \textbf{(3.9c)}$$

where $c(x) = 0$ is a vector of nonlinear constraints, x_L is a vector of lower bounds for x and x_U is a vector of upper bounds for x. Therefore, the first order KKT conditions for (3.9) can be written as

$$\nabla_x L(x^*, u^*, v^*) = \nabla f(x^*) + \nabla c\left(x_\mu^*\right)v^* - u_L^* + u_U^* = 0, \quad \textbf{(3.10a)}$$
$$c(x^*) = 0, \quad \textbf{(3.10b)}$$
$$0 \le u_L^* \perp (x^* - x_L) \ge 0, \quad \textbf{(3.10c)}$$
$$0 \le u_U^* \perp (x_U - x^*) \ge 0, \quad \textbf{(3.10d)}$$

where u_L^* and u_U^* are vectors of KKT multipliers for lower and upper bounds, respectively. The notation $y \perp z$ denotes that $y_i = 0$ (inclusive) or $z_i = 0$ for all elements i of these vectors.

Consider a NLP of the form presented in (3.9). Sets of indices of active bounds at a local solution x^* are defined as $A_L = \{i_1^L, i_2^L, \dots, i_j^L\}, j = 1, \dots, |A_L|$ and $A_U = \{i_1^U, i_2^U, \dots, i_j^U\}, j = 1, \dots, |A_U|$ with active multipliers u_{A_L} and u_{A_U} that correspond to these sets, for lower and upper bounds, respectively. For $j = 1, \dots, \left|A_{L\backslash U}\right|$ and $i = 1, \dots, |x|$, matrices E_L and E_U, that determine the corresponding variables at bounds, consist of elements

$$\left\{E_{L\backslash U}\right\}_{ij} = \begin{cases} 1 & \text{if } i = i_j^{L\backslash U}, \\ 0 & \text{otherwise,} \end{cases} \quad \textbf{(3.11)}$$

where the index notation $L\backslash U$ means that either L or U is used. By assuming that the active bounds are known, the KKT conditions for the above described system can be presented as

$$\nabla_x L\left(x, u_{A_L}, u_{A_U}, v\right) = \nabla f(x) + \nabla c(x)v - E_L u_{A_L} + E_U u_{A_U} = 0, \quad \textbf{(3.12a)}$$
$$c(x) = 0, \quad \textbf{(3.12b)}$$
$$E_U^T x = E_U^T x_U, \quad \textbf{(3.12c)}$$
$$E_L^T x = E_L^T x_L. \quad \textbf{(3.12d)}$$

By applying Newton's method to the KKT conditions (3.12) the following linear system is achieved:

$$\begin{bmatrix} W^k & \nabla c(x^k) & -E_L & E_U \\ \nabla c(x^k)^T & 0 & 0 & 0 \\ -E_L^T & 0 & 0 & 0 \\ E_U^T & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_v \\ p_{u_{A_L}} \\ p_{u_{A_U}} \end{bmatrix} = - \begin{bmatrix} \nabla_x L(x^k, u_{A_L}^k, u_{A_U}^k, v^k) \\ c(x^k) \\ E_L^T(x_L - x^k) \\ E_U^T(x^k - x_U) \end{bmatrix}, \quad \textbf{(3.13)}$$

where $W^k = \nabla_x^2 L(x^k, u_{A_L}^k, u_{A_U}^k, v^k)$. The system (3.13) corresponds to the first-order KKT conditions of the following QP problem:

$$\min_{p_x} f(x^k) + \nabla f(x^k) p_x + \frac{1}{2} p_x^T W^k p_x \quad \textbf{(3.14a)}$$

s.t.

$$\nabla c(x^k)^T p_x + c(x^k) = 0, \quad \textbf{(3.14b)}$$
$$E_U^T(x^k + p_x) = E_U^T x_U, \quad \textbf{(3.14c)}$$
$$E_L^T(x^k + p_x) = E_L^T x_L. \quad \textbf{(3.14d)}$$

The above QP problem is, in fact, the quadratic subproblem of the original NLP (3.9) and the solution of the subproblem can be used to obtain the next trial solution as $x^{k+1} = x^k + p_x$. Additionally, the quadratic programming multipliers, which determine the likely active set for each iteration, are also updated as $(u_{A_L}^{k+1}, u_{A_U}^{k+1}, v^{k+1}) = (u_{A_L}^k + p_{u_{A_L}}, u_{A_U}^k + p_{u_{A_U}}, v^k + p_v)$. [5 pp. 135-136].

A basic strategy for solving a constrained convex NLP problem can be defined as follows [5 p. 141] [10 p. 472]:

Choose convergence limits $\epsilon_1 > 0$, $\epsilon_2 > 0$ and initial values x^0, v^0, $u_{A_L}^0$, $u_{A_L}^0$.

For $k \geq 0$, while $||p_x|| > \epsilon_1$ and $\max\left(\left|\left|\nabla_x L(x^k, u_{A_L}^k, u_{A_U}^k, v^k)\right|\right|, \left|\left|c(x^k)\right|\right|\right) > \epsilon_2$:

1. Check bounds $x_L \leq x \leq x_U$
    a. Add new active constraints to A_L and A_U if any
2. Build matrices E_L and E_U
3. Compute $\nabla_x L(x^k, u_{A_L}^k, u_{A_U}^k, v^k)$, W^k, $c(x^k)$, $\nabla c(x^k)$
4. Ensure the positive definiteness of W^k and nonsingularity of the KKT matrix
5. Solve the QP subproblem using (3.13)

6. If $p_x = 0$ then check $\left(u_{A_L}^k + p_{u_{A_L}}, u_{A_U}^k + p_{u_{A_U}}\right) \geq 0$
    a. If true then x^k is a solution, STOP
    b. Else remove the variable with the most negative multiplier $u_{A_L}^k, u_{A_U}^k$ from its active set, update E_L and E_U, and go to step 3

    Else ($*p_x \neq 0*$)
    a. For each inactive constraint i:
        i. If $x_{L_{(i)}} > x_{(i)}^k + p_{(i)}^k$
            i. $\gamma_j = \frac{x_{L_{(i)}} - x_{(i)}^k}{p_{(i)}^k},\ j = 0,1,2,\ldots$
        ii. Else if $x_{U_{(i)}} < x_{(i)}^k + p_{(i)}^k$
            i. $\gamma_j = \frac{x_{U_{(i)}} - x_{(i)}^k}{p_{(i)}^k},\ j = 0,1,2,\ldots$
    b. Choose step size: $\alpha^k = \min(1, \min \gamma) \in [0,1]$
7. Update the iterates
    a. $x^{k+1} = x^k + \alpha^k p_x$
    b. $v^{k+1} = v^k + \alpha^k p_v$
    c. $u_{A_L}^{k+1} = u_{A_L}^k + \alpha^k p_{u_{A_L}}$
    d. $u_{A_U}^{k+1} = u_{A_U}^k + \alpha^k p_{u_{A_U}}$

As the presented strategy is based on the Newton's method, the same Hessian modification and global convergence strategies mentioned in Chapter 3.1 can be applied.

3.3 Interior Point Methods

Interior Point (IP) methods (also known as barrier methods) try to find a feasible solution to a nonlinear programming problem by taking steps through the interior of the feasible region using gradient information. In contrast, the well-known simplex algorithm for LP problems takes steps around the boundary of the region, rather than through it. The first interior point method was proposed by mathematician John von Neumann in 1948 in discussion with the inventor of simplex algorithm George Dantzig. [11 pp. 67, 70].

Still, IP methods weren't popular after the 1960's because of the dominance of simplex methods in linear programming and augmented Lagrangian and sequential quadratic programming (SQP) methods in nonlinear programming. However, in 1984, a paper published by Narendra Karmarkar changed the landscape by introducing a fast polynomial-time interior point method for linear programming, which he claimed to be 50 times faster than the simplex method. In the following year, a formal equivalence between Karmarkar's methods and the logarithmic barrier interior point method was presented. This revelation opened the doors for researchers to further explore the possibilities of interior point methods for solving LPs and NLPs. [12].

In the following chapter a basic version of the primal-dual logarithmic barrier method implemented in IPOPT optimization software package is covered. The algorithm has proven to be very efficient in large-scale optimization benchmark tests compared to other large-scale NLP solvers [5 pp. 171-175]. Further details about the original method can be found in [13].

3.4 Primal-Dual Logarithmic Barrier Method

The general idea of barrier function methods is to introduce a barrier term which will replace the inequality constraints by forming a new objective function including the constraints in the barrier term. The principal is very similar to the Lagrangian relaxation method. The barrier term is a function that approaches $+\infty$ as any feasible interior point approaches the boundary of the feasible region. This penalty averts the iterations from going into infeasible regions which then allows the relaxation of the complementary conditions. As the penalty increases in the vicinity of the region boundaries, a dynamic weight is introduced to the penalty to allow the iterations to reach the region boundaries where solutions are generally found. [11 p. 128]. In this chapter we will focus on the logarithmic barrier function.

Consider a NLP of the following form:

$$\min f(x) \tag{3.15a}$$

s.t.

$$c(x) = 0, \tag{3.15b}$$

$$x \geq 0. \tag{3.15c}$$

This modified form of (3.9) is used to simplify the derivation; however the interior point method is applicable to other NLP formulations as well including (2.1) and (3.9). By introducing the logarithmic barrier function for (3.15), the NLP is transformed to [5 p. 151]

$$B(x, \mu) = f(x) - \mu \sum_{i=1}^{m} \ln(x_i), \qquad \mu > 0, \tag{3.16a}$$

s.t.

$$c(x) = 0, \tag{3.16b}$$

$$x > 0. \tag{3.16c}$$

For future references, $B(x, \mu)$ with fixed μ is henceforth denoted as $B_\mu(x)$.

The interior point strategy forces the values of x to stay strictly inside the feasible region since the logarithmic barrier function is not defined when $x < 0$ and becomes unbounded at $x = 0$. Now, if $x(\mu)$ is considered as an unconstrained minimizer of $B(x, \mu)$, then under mild conditions

$$\lim_{\mu \to 0+} x(\mu) = x^*, \tag{3.17}$$

where x^* is a local minimizer of (3.15) [11 p. 129]. In other words, the solution of the barrier problem (3.16) approaches the solution of the original NLP problem (3.15) when $\mu \to 0^+$. More information and theoretical proofs of this can be found in [14].

Now, if $c(x) = 0$ at $x^*(\mu) = x_\mu^*,\ \mu > 0$ fulfils the LICQ, then the first order optimality conditions hold for the solution of the barrier problem and are defined as follows:

$$\nabla f(x_\mu^*) + \nabla c(x_\mu^*) v - \mu X^{-1} e = 0, \tag{3.18a}$$
$$c(x_\mu^*) = 0, \tag{3.18b}$$

where $X = diag\{x\}$, $e = [1,1, \dots ,1]^T$, and the solution vector $x_\mu^* > 0$. With these conditions Newton-based methods can be applied to solve the NLP problem.

A closer analysis of the barrier function tells us that the gradients of the barrier function are unbounded at the constraint boundary, which in turn indicates very steep and ill-conditioned response surfaces for the barrier function. Furthermore, the extreme nonlinearity of the barrier function makes the direct solution difficult to obtain. Therefore, some changes to the system should be applied to ease the nonlinearity. One effective approach is to modify the first order optimality conditions to form a primal-dual system, which introduces new dual variables u along with the equation $Xu = \mu e$ to present the optimality conditions as follows:

$$\nabla f(x_\mu^*) + \nabla c(x_\mu^*) v - u = 0, \tag{3.19a}$$
$$Xu = \mu e, \tag{3.19b}$$
$$c(x_\mu^*) = 0, \tag{3.19c}$$

where the barrier multipliers u correspond to the KKT multipliers for the bound constraints as $\mu \to 0$. The substitution eases the nonlinearity and, in fact, can be interpreted as the relaxation of the complementary KKT conditions for (3.15). Another approach that helps to deal with the ill-conditioned problem is the μ update strategy, which is discussed later. Now a Newton-based strategy can be applied to solve the problem via the primal-dual system. [5 p. 153].

An error is defined for the optimality conditions of the problem as [5 p. 153]

$$E_\mu(x, v, u) := \max\{\left|\left|\nabla f\ (x) + \nabla c(x)v - u\right|\right|_\infty, \left|\left|c(x)\right|\right|_\infty, \left|\left|Xu - \mu e\right|\right|_\infty\}. \quad \textbf{(3.20)}$$

The error E_0 represents the optimality error for the original problem (as in $\mu = 0$) and is used to terminate the overall algorithm when the user-defined error tolerance limit ϵ_{tol} is satisfied. Likewise, the barrier problem is solved for a tolerance determined by $E_{\mu_l} \leq \kappa_\epsilon \mu_l$, where $\kappa_\epsilon > 0$ is the constant accuracy control parameter.

To solve the barrier problem a nested approach can be used. First, we form an outer loop that is used to calculate a value for μ. Then μ is assumed fixed and the barrier problem is solved in the inner loop. Generally, choosing μ is a trade-off between robustness and efficiency. The smaller the μ updates are the more robust the iterations become since the starting points are closer to the solution, but then again more iterations are required to achieve the optimum for the original problem.

Following μ update strategy, capable of decreasing μ at a superlinear rate, is presented by Wächter and Biegler [13] for the interior point method:

$$\mu_{l+1} = \max\left\{\frac{\epsilon_{tol}}{10}, \min\left\{\kappa_\mu \mu_l, \mu_l^{\theta_\mu}\right\}\right\}. \quad \textbf{(3.21)}$$

Wächter and Biegler [13] have determined through practice that good general values for the above constants are $\kappa_\epsilon = 10$, $\kappa_\mu = 0.2$, and $\theta_\mu = 1.5$. A lower limit of $\frac{\epsilon_{tol}}{10}$ for μ_l is introduced to keep the values from becoming too small since it might lead to numerical difficulties in the computation of the inner iteration loop.

The presented update strategy for μ_l is simple and easy to implement. However, other more sophisticated parameter updating strategies, which deal better with ill-conditioned problems, do exist. Such strategies include the predictor-corrector method for solving LPs and QPs and an adaptive μ strategy for NLPs. These methods can update the barrier parameter simultaneously with primal and dual variables. [5 p. 154].

Finally, to solve the barrier problem, a Newton-based method with line search is presented. In this approach the search directions $p^k = (d_x^k, d_v^k, d_u^k)$ for the primal and dual variables are calculated from the following linearization of the primal-dual system [5 p. 154]:

$$\begin{bmatrix} W^k & \nabla c(x^k) & -I \\ \nabla c\left(x^k\right)^T & 0 & 0 \\ U^k & 0 & X^k \end{bmatrix} \begin{bmatrix} d_x^k \\ d_v^k \\ d_u^k \end{bmatrix} = - \begin{bmatrix} \nabla f\left(x^k\right) + \nabla c\left(x^k\right) v^k - u^k \\ c\left(x^k\right) \\ X^k u^k - \mu_l e \end{bmatrix}, \qquad \textbf{(3.22)}$$

where $W^k = \nabla_x^2 L(x^k, v^k)$ (or an approximation), $U^k = diag\{u^k\}$, $X^k = diag\{x^k\}$ and $x^k, u^k > 0$. Moreover, the problem can be simplified by eliminating the last block row and solving the smaller, symmetric linear system

$$\begin{bmatrix} W^k + \Sigma^k & \nabla c(x^k) \\ \nabla c\left(x^k\right)^T & 0 \end{bmatrix} \begin{bmatrix} d_x^k \\ d_v^k \end{bmatrix} = - \begin{bmatrix} \nabla B_\mu\left(x^k\right) + \nabla c\left(x^k\right) v^k \\ c\left(x^k\right) \end{bmatrix}, \qquad \textbf{(3.23)}$$

where $\Sigma^k \coloneqq \left(X^k\right)^{-1} U^k$ [5 p. 154]. The vector d_u^k is then obtained from [5 p. 155]

$$d_u^k = \mu_l \left(X^k\right)^{-1} e - u^k - \Sigma^k d_x^k. \qquad \textbf{(3.24)}$$

A straight forward solution of the linear system (3.23) would be the preferred technique but it is not always possible since the matrix $W^k + \Sigma^k$ is required to be positive definite to guarantee that the search direction is a descent direction [15]. Moreover, rank deficiency of the constraint Jacobian causes the KKT matrix to be singular, which might lead to an unsolvable system. To counter these issues an inertia correction procedure can be applied [13].

Then a line search operation can be applied with step size $\alpha^k, \alpha_u^k \in (0,1]$ to attain the next trial solution as [5 p. 155]

$$x^{k+1} \coloneqq x^k + \alpha^k d_x^k, \quad \textbf{(3.25a)}$$
$$v^{k+1} \coloneqq v^k + \alpha^k d_v^k, \quad \textbf{(3.25b)}$$
$$u^{k+1} \coloneqq u^k + \alpha_u^k d_u^k. \quad \textbf{(3.25c)}$$

Due to the nature of the logarithmic barrier function, the barrier problem is defined only at positive values of x. To maintain this condition, the step-to-the-boundary rule is applied:

$$\alpha_{max}^k \coloneqq \max\{\alpha \in (0,1]: x^k + \alpha d_x^k \geq (1-\tau_l)x^k\}, \quad \textbf{(3.26a)}$$
$$\alpha_u^k \coloneqq \max\{\alpha \in (0,1]: u^k + \alpha d_u^k \geq (1-\tau_l)u^k\} \quad \textbf{(3.26b)}$$

with the parameter

$$\tau_l = \max\{\tau_{min}, 1-\mu_l\}, \quad \textbf{(3.27)}$$

where $\tau_{min} \in (0,1)$ is the minimum value for τ_l. Common values for τ_{min} are in proximity of 1, for example 0.99. The step size $\alpha^k \in (0, \alpha_{max}^k]$ for x and v can be calculated using a backtracking line search algorithm that tests a decreasing sequence of trial step sizes $\alpha^{k,j} = 2^{-j}\alpha_{max}^k$ (with $j = 0,1,2,\ldots$) until a proper step size is found. [5 p. 155].

The basis of a backtracking line search algorithm is that the chosen step size fulfils the Armijo condition:

$$f(x^k + \alpha^k p^k) \leq f(x^k) + \eta\alpha^k \nabla f\left(x^k\right)^T p^k, \quad \textbf{(3.28)}$$

where $\eta \in \left(0, \frac{1}{2}\right]$ is a search control parameter. In short, the Armijo condition tests that the objective function decreases sufficiently for the chosen step size. Additional conditions, including the Wolfe and the Goldstein conditions, are commonly applied to ensure the convergence of the solution. [5 pp. 47-48].

Finally, a nested approach is presented for the solving strategy of the primal-dual barrier method. The following algorithm is adapted from [13]:

Choose constants $\epsilon_{tol} > 0, \kappa_\epsilon > 0, \kappa_\mu \in (0,1), \theta_\mu \in (1,2), \tau_{min} \in (0,1)$ and $\eta \in \left(0, \frac{1}{2}\right]$. Define initial values for starting points $x^0 > 0$, v^0, u^0, and for the barrier parameter $\mu_0 > 0$. Iteration counters are initialized as $k = 0$ and $l = 0$.

1. Check convergence of the overall problem. If $E_0\left(x^k, v^k, u^k\right) \leq \epsilon_{tol}$, then STOP (converged).
2. Check convergence of the barrier problem. If $E_{\mu_l}\left(x^k, v^k, u^k\right) \leq \kappa_\epsilon \mu_l$, then:
    a. Compute μ_{l+1} and τ_l from (3.21) and (3.27), and set $l = l + 1$
3. Compute the search direction. Calculate (d_x^k, d_v^k, d_u^k) from (3.23) and (3.24).
4. Backtracking line search.
    a. Initialize the line search. Set $\alpha^{k,0} = \alpha_{max}^k$ with α_{max}^k from (3.26a), and set $j = 0$.
    b. Compute new trial point. Set $x^k(\alpha^{k,j}) \coloneqq x^k + \alpha^{k,j} d_x^k$.
    c. Check sufficient decrease with respect to the current iterate. If Armijo condition (3.28) fails, then:
        i. Increase $j = j + 1$ and calculate a new step length from $\alpha^{k,j} = 2^{-j}\alpha_{max}^k$. Go back to 4b.
5. Accept the trial point. Set $\alpha^k \coloneqq \alpha^{k,j}$ and calculate α_u^k from (3.26b). Update the estimates as in (3.25).
6. Continue with the next iteration. Update iteration counter $k = k + 1$ and continue from step 1.

The proposed algorithm can be augmented with several improvements, as is presented in the original article [13]. The original algorithm features a line search filter method that uses constraint violation in addition to the objective function in order to determine if the chosen step provides sufficient progress. Furthermore, a switching condition is introduced to prevent the algorithm from approaching feasible but non-optimal points. The filtering procedure, on the other hand, discards those trial points that increase the constraint violation over a certain threshold and inhibits the algorithm from cycling back

to those already rejected points. With these restrictions, finding an acceptable trial point is not always possible, therefore a feasibility restoration phase is introduced. The restoration algorithm tries to find a new acceptable trial point by reducing the constraint violation with an iterative method. Moreover, to reduce the number of rejected trial points, a technique known as second-order correction is applied. The method tries to reduce the infeasibility of the trial point by applying an additional Newton-type step for the constraints at the trial point using the constraint Jacobian of the current iterate. Comparatively, instead of the line search filter method, a trust region strategy can be used for achieving global convergence. Biegler presents an adaption of the interior point algorithm with a trust region method in [5 pp. 158-160].

3.5 Nonlinear Programming Software

Numerous NLP software packages have been developed over the course of decades that utilize a wide range of methods which each have their uses. No "best-in-everything" algorithm exists but methods should be carefully chosen depending on the characteristics of the problem to achieve the best performance. A great source of information about different algorithms and software packages can be found on NEOS server website [16] which offers free internet-based optimization services for solving numerical optimization problems and runs over 60 different state-of-the-art solvers to choose from. Another great site is the Decision Tree for Optimization Software run by Professor Hans Mittelmann [17]. The site offers a guide for choosing proper algorithms for your optimization problems including an impressive list of optimization software. Additionally, the site has collected a pool of benchmark tests to compare the performance of the algorithms in reality.

In this chapter, a few of the most well-known NLP optimization software packages are described, implementing SQP and interior point algorithms. Additionally, an optimal control problem benchmark test is presented.

These software packages were chosen because they are well-known, robust, efficient and tested through-and-through in numerous applications. SNOPT and KNITRO offer trial versions for evaluation purposes but SOCS requires a commercial license. IPOPT, on the other hand, is available for free of charge under the Eclipse Public License even for commercial purposes.

3.5.1 Sequential Quadratic Programming Solvers

SNOPT [18]: A large-scale NLP solver developed by Gill, Murray and Saunders [19] in Fortran 77+. The software utilizes a sparse SQP algorithm with limited-memory quasi-Newton approximations to the Hessian of Lagrangian. The algorithm is designed for nonlinear problems with expensive functions and gradients (uses only first derivatives). The functions are assumed smooth but are not required to be convex. Global convergence is ensured with an augmented Lagrangian merit function and infeasible problems are handled with elastic bounds on constraints. The software has been used in many applications in fields of trajectory optimization, optimal control, robotics, engineering design, nonlinear networks, trade models, portfolio analysis and spatial equilibrium [20]. Some of the recent articles featuring SNOPT include applications in renewable energy management [21], descent trajectory optimization [22] and optimizing design of lithium-ion battery pack [23].

SOCS [24]: A sparse SQP algorithm developed at Boeing by Betts and coworkers in Fortran 77 [25]. This algorithm is able to utilize exact second order derivatives but also incorporates quasi-Newton approximations (SR1, BFGS and SSQN). The positive definiteness of the projected Hessian is ensured with the Levenberg modification strategy. The QP subproblem can be solved using a sparse Schur-complement method or a dense nullspace QP. An augmented Lagrangian line search procedure is used to achieve global convergence. The package also incorporates sparse and dense primal-dual interior point solvers with a nonlinear filter globalization method. The software package also implements special optimal control subroutine package that utilizes a direct transcription or collocation method to convert the continuous control problem

into a discrete approximation that can be solved with the NLP solver. Applications for this software include trajectory optimization, chemical process control and machine tool path definition [26]. SOCS has been featured in recent optimal control articles concerning vehicle control [27], optimal control software comparison [28] and mesh refinement in optimal control [29].

3.5.2 Interior Point Solvers

IPOPT [30]: A software package for large-scale nonlinear optimization originally developed by Wächter and Biegler [13] in Fortran 77 but has now been rewritten in C++. The NLP solver uses primal-dual interior point algorithm with a line search filter method and includes inertia corrections for the KKT matrix, second-order corrections and the feasibility restoration phase for the filter method. The algorithm utilized exact second derivatives but also includes BFGS and SR1 quasi-Newton approximation strategies. Other features include a warm starting strategy utilizing primal and dual variables and multiple barrier parameter updating strategies such as Mehtrotra's algorithm, the monotone Fiacco-McCormick approach and adaptive strategies. IPOPT has been used in numerous applications including optimal control, asset allocation, portfolio optimization, risk management, parameter identification, nonlinear scaling, unsupervised learning, process simulators, data reconciliation and robotics [31]. It is also included in General Algebraic Modeling System (GAMS) mathematical programming environment. Recent articles employing IPOPT feature challenges like real-time optimization of reverse osmosis networks [32], simultaneous optimal design and control [33] and dynamic parameter estimation [34].

KNITRO [35]: An optimization software package developed by Byrd, Nocedal and Waltz in C programming language [36]. The software is primarily designed for solving large-scale NLP problems but can also solve mixed integer problems. The newest version of the package incorporates four different algorithms including two interior point methods, an active set method and a SQP method. The solver can choose between the algorithms automatically or the user can choose which one is utilized. The software implements

two primal-dual interior point methods that solve the KKT matrix either by direct linear algebra (Interior/Direct) or utilizing projected conjugate gradient iterations (Interior/CG). The active set method forms a quadratic model of the original problem and utilizes a sequential linear quadratic programming algorithm that uses LP subproblems to estimate the active set. The SQP method is also an active set method but it utilizes QP subproblems to find a solution, therefore requiring the least amount of function/gradient evaluations. KNITRO has been utilized in many different application areas including finance and banking, computational economics and game theory, statistics and data analysis, energy, sustainable development, optimal control and dynamic optimization, telecommunication, optics and spectroscopy, mathematics and geometry. Typical uses and literature references can be found in [37]. The most recent articles featuring KNITRO include kinematical optimization of a locomotion strategy [38], power flow optimization [39] and optimal control problem to reduce aircraft noise [40].

3.5.3 Performance Trends

The optimal control problem presented here is based on a nonlinear continuously stirred tank reactor originally introduced by Hicks and Ray [41]. For the purpose of this benchmark the differential control problem is transformed into the following NLP problem by applying a backward differentiation formula:

$$\min \sum_{i=1}^{N} [\alpha_1 (C_{des} - C_i)^2 + \alpha_2 (T_{des} - T_i)^2 + \alpha_3 (u_{des} - u_i)^2]/N \qquad \textbf{(3.29a)}$$

s.t.

$$C_{i+1} = C_i + \tau \bar{C}_{i+1} N^{-1}, \qquad \textbf{(3.29b)}$$

$$T_{i+1} = T_i + \tau \bar{T}_{i+1} N^{-1}, \qquad \textbf{(3.29c)}$$

$$\bar{C}_i = (1 - C_i)\theta^{-1} - k_{10} \exp(\omega_i)\, C_i, \qquad \textbf{(3.29d)}$$

$$\bar{T}_i = (T_f - T_i)\theta^{-1} + k_{10} \exp(\omega_i)\, C_i - \alpha u_i (T_i - T_c), \qquad \textbf{(3.29e)}$$

$$\omega_i T_i + \eta = 0, \qquad \textbf{(3.29f)}$$

$$C_1 = C_{init}, \qquad T_1 = T_{init}, \qquad \textbf{(3.29g)}$$

$$C_i \in [0,1], \qquad T_i \in [0,1], \qquad u_i \in [0,500], \qquad \textbf{(3.29h)}$$

where the number of variables and equations are given by $n = 6N - 2$ and $m = 5N - 2$, respectively. The equations (3.29b-c) correspond to the concentration and temperature states of the tank reactor, and equations (3.29d-e) represent the differential dynamics of those states. The temperature dependence of reaction rate is handled as a separate equality constraint (3.29f) to ease the nonlinearity.

The NLP problem and following results are derived and gathered by Biegler [5 pp. 172-173].

The tests were conducted using parameter values

- $C_{init} = 0.1367$ (Concentration initial value)
- $T_{init} = 0.7293$ (Temperature initial value)
- $C_{des} = 0.0944$ (Concentration target value)
- $T_{des} = 0.7766$ (Temperature target value)
- $u_{des} = 340$ (Coolant flow rate target value)
- $\alpha = 1.95 * 10^{-4}$ (Heat transfer area)
- $\alpha_1 = 106$ (Tuning parameter for concentration)
- $\alpha_2 = 2000$ (Tuning parameter for temperature)
- $\alpha_3 = 0.001$ (Tuning parameter for coolant flow)
- $k_{10} = 300$ (Reaction rate coefficient)
- $\eta = 1$ (Ratio of activation energy and gas constant)
- $\theta = 20$ (Volume/Flow ratio)
- $T_f = 0.3947$ (Feed temperature)
- $T_c = 0.3816$ (Coolant temperature)
- $\tau = 10$ (Backward differentiation step size)

with variables initialized by infeasible points

- $\bar{C}_i = 1$ (Concentration change)
- $\bar{T}_i = 1$ (Temperature change)
- $C_i = C_{init} + (C_{des} - C_{init})(i-1)N^{-1}$
- $T_i = T_{init} + (T_{des} - T_{init})(i-1)N^{-1}$
- $\omega_i = \eta$

- $u_i = 250.$ (Coolant flow rate)

The tests were run on IPOPT (version 3.5), KNITRO (version 5.2.0) and SNOPT (version 7.2-4) using default settings of each solver. The used hardware was a PC running Windows XP operating system with 2.4 GHz Intel Core2 Duo processor and 2GB RAM. Running the test problem with increasing values of scaling parameter N gave results presented in Table 3.1.

Table 3.1. Results of the scalable optimal control problem in a format of "(Iterations)/(Minor iterations) [CPU seconds]". The number of variables and equations are $n = 6N - 2$ and $m = 5N - 2$.

N	IPOPT	KNITRO	SNOPT
5	9 [0.016]	16/0 [0.046]	13/33 [0.063]
10	9 [0.016]	19/0 [0.093]	15/55 [0.063]
50	9 [0.032]	20/0 [0.078]	15/357 [0.11]
100	9 [0.11]	18/0 [0.109]	13/603 [0.156]
500	9 [0.64]	96/337 [4.062]	23/5539 [4.828]
1000	9 [1.422]	116/771 [16.434]	31/10093 [13.734]

The test results present some trends to the software performance when the size of the problem grows but as Biegler stated in his text, this is not a definitive performance comparison since the results are strongly dependent on the hardware environment, operating system and modelling environment.

From the results in Table 3.1 it can be clearly observed that the number of iterations stay constant for IPOPT and this is due to the Newton-based solving method of the primal-dual equations that utilizes a sparse matrix solver and a filter line search to promote larger steps. Also for the large values of N (500, 1000) the CPU time appears to be approximately linear and much faster than for the other solvers. KNITRO also uses a similar Newton-based solving method of the primal-dual equations in the direct mode but with a different line search strategy which accounts for the difference in iterations to IPOPT up to $N = 100$. For larger values of N KNITRO automatically switches to the conjugate gradient version of the algorithm to provide more careful but computationally expensive trust region steps. As for SNOPT, the number of iterations increase only

slightly with N but because of the BFGS updates, that are exploited because the solver doesn't use exact second derivatives, and heavily constrained QPs, the CPU time increases heavily with N and the solver is slower than the other two. Although, in the case of $N = 1000$, SNOPT does perform better than KNITRO since the number of minor iterations increased relatively less with the increase in N for SNOPT.

3.6 Summary

The choice between interior point and active set SQP methods is related to the details of the optimization problem. Especially interesting characteristics are the size of the problem and number of constraints and structure of the matrices.

SQP methods perform best when the number of active constraints does not differ much from the total number of variables. They tend to be more robust on badly scaled problems than the nonlinear interior point methods. [10 p. 560]. The SQP methods also deal well with problems having significant nonlinearities in the constraints [10 p. 529].

Interior point methods review every constraint in the optimization problem during each of the iterations, even if some of the constraints are inactive at the solution. Consequently, the cost of the primal-dual iteration can become extreme in some applications. [10 p. 593]. Additionally, interior point methods may require more iterations than active set methods to solve the optimization problem, since the inner loop barrier problem is solved for multiple values of μ. On the other hand, active set methods require solving the more expensive QP subproblem. Thus, active set methods are favored when there are only few inequality constraints or an estimate of the solution available (known as warm starting) and the solving of the subproblem is not expensive. [4]. However, warm starting capabilities are also available for interior point methods, and for example IPOPT can utilize the values of primal and dual variables from a previous solution of a related problem as an initial guess [42].

Interior point methods, on the other hand, are often faster than SQP methods in solving large problems with many inequality constraints as they avoid the combinatorial

problem of selecting the active set. Additionally, interior point methods more readily exploit the sparsity and structure in the KKT conditions since the nonzero structures in the matrixes do not change over iterations, unlike in the active set methods. These features are important especially for large-scale optimization problems and when a large number of bounds are active. [4]. As problem size and number of constraints have less of an effect on interior point methods, they are often favored over SQP methods in large-scale optimization.

A detailed review of large-scale nonlinear optimization methods can be found in [43] by Gould et al. The review considers active set and interior point methods and includes software examples implementing these.

4 Model Predictive Control

Model Predictive Control (MPC) is an advanced multivariate control strategy that utilizes internal dynamic models to predict the future states of the process in order to calculate an optimal control sequence that minimizes the constrained objective function. MPC exploits the so-called receding horizon control strategy, where the optimal control sequence is predicted for a user-defined number of steps (prediction horizon) into the future at each control cycle but only the first step is executed. Hence, the prediction horizon shifts towards the future for each new iteration of the control problem, and the optimal control sequence is adjusted to possible process disturbances and changes.

MPC strategies have been used in chemical industry since 1970s [1] and it has become the de facto technology in the oil industry [44]. Survey made by Qin and Badgwell [45] in 2003 estimated over 4500 industrial MPC applications with almost 2000 in refining industry and 550 in petrochemical industry and close to 150 in chemical industry. Other areas where MPC technology was utilized in lesser degree were pulp and paper, air and gas, utility, mining and metallurgy, food processing, polymer, furnaces, aerospace and defense and automotive industries. A more recent survey (published in 2008) by Bauer and Craig [46] regarding economic assessment of advanced process control (APC) reveals that MPC and constraint control are being the leading control strategies utilized in APC projects. The number of APC applications was estimated to be approximately 6000 in 2005 [47].

MPC applications commonly use linear models since they allow the use of powerful matrix algebra operations making the calculations fast and robust. The linear models usually provide good enough controller performance when the process states stay around the linearization point and measurement sampling periods are short enough. In more dynamic processes or during process events, where linear models might prove to be insufficiently accurate, nonlinear MPC strategies should be applied together with nonlinear models. [5 p. 12]. More information about nonlinear MPC can be found in an article by Allgöwer, Findeisen and Nagy (2004) [48] introducing the basic principles of nonlinear MPC and in a more recent book by Grüne and Pannek (2011) [49]. Additionally, interesting "hybrid" nonlinear MPC strategies that utilize Wiener and

Hammerstein models have been studied and the results are promising in the control of chemical processes [50], [51], [52]. The main advantage is that the computational burden is heavily reduced compared to a fully nonlinear MPC application.

The control optimization problem is usually modelled as minimization of a quadratic objective function. The objective function can include different objectives that are sought to be minimized but the primary objective is to minimize the errors between predicted states and target trajectories. This drives the process towards the target trajectory in a manner (aggressive - conservative) determined by the internal relativity of the terms (affected by weighting and scaling) in the objective function while respecting the given constraints. Additionally, the objective function might include penalties on large input values and input changes. These are introduced to regulate the aggressiveness of the control since too large and frequent control moves can make the process unstable especially if the changes transition the process out of the model validity zone. The controller is tuned using scaling and weights associated with each variable to find a balance that induces the desired behavior of the process. The main idea of the online control optimization is that the best control sequence at each control cycle is found and the control behavior is adjusted so that robust and efficient control of the process can be achieved.

A linear time invariant model describing the process dynamics can be presented in a following state-space form:

$$x^{k+1} = Ax^k + Bu^k, \quad \textbf{(4.1a)}$$
$$y^k = Cx^k + d^k, \quad \textbf{(4.1b)}$$

where x^k is the vector of state variables, u^k is the vector of manipulated variables, A is the state-space model matrix, B is the manipulated variable model matrix, y^k is the vector of output variables, C is the controlled variable mapping matrix and d^k is the vector of known disturbances. This discrete presentation is related to continuous time by $t^k = t_0 + k\Delta t$, where t_0 is the current time and Δt is the sampling time. [5 pp. 12-13].

For the system described in (4.1) the quadratic optimization problem that minimizes controlled variable offset and manipulated variable deviations can be defined as

$$\min_{u^k, x^k, y^k} \sum_{k=1}^{H_p} \left(y^k - y^{sp}\right)^T Q_y \left(y^k - y^{sp}\right) + \sum_{k=1}^{H_u} \left(u^k - u^{k-1}\right)^T Q_u \left(u^k - u^{k-1}\right) \quad \textbf{(4.2a)}$$

s.t.

$$x^k = Ax^{k-1} + Bu^{k-1} \quad \text{for } k = 1, \dots, H_p, \quad \textbf{(4.2b)}$$

$$y^k = Cx^k + d^k \quad \text{for } k = 1, \dots, H_p, \quad \textbf{(4.2c)}$$

$$u^k = u^{H_u} \quad \text{for } k = H_u + 1, \dots, H_p, \quad \textbf{(4.2d)}$$

$$-\hat{b} \leq \hat{A}^T u^k \leq +\hat{b} \quad \text{for } k = 1, \dots, H_p, \quad \textbf{(4.2e)}$$

$$u^L \leq u_k \leq u^U \quad \text{for } k = 1, \dots, H_p, \quad \textbf{(4.2f)}$$

$$-\Delta u^{max} \leq u^k - u^{k-1} \leq +\Delta u^{max} \quad \text{for } k = 1, \dots, H_p, \quad \textbf{(4.2g)}$$

where H_p is the output prediction horizon, H_u (s.t. $1 \leq H_u \leq H_p$) is the input prediction horizon, y^{sp} is the target trajectory for output variables, Q_y is the diagonal penalty matrix for output variables, Q_u is the diagonal penalty matrix for manipulated variables, $\hat{b}$ are the bounds for state constraints, $\hat{A}$ is the transformation matrix for state constraints, and Δu^{max} is the maximum allowed change for the actuators. Note that the prediction horizon is divided into input and output prediction horizons so that the length can be adjusted separately for both. The equality constraints (4.2b) and (4.2c) present the state-space model of the process dynamics. Constraint (4.2d) is used to hold the inputs constant for the remainder of the prediction horizon if the input prediction horizon is shorter than the output prediction horizon. The input values are ensured to respect the state constraints by (4.2e). Constraint (4.2f) introduces bounds on minimum and maximum values of the inputs to prevent the saturation of actuators. Finally, constraint (4.2g) provides the bounds to control the input deviations and adjusts the aggressiveness of the control movements.

The input prediction horizon is usually shorter than the output prediction horizon since only the first control move is executed and the later control moves have less of an effect on the optimization task. On the other hand, the output prediction horizon should be long enough to allow the simulation to reach a steady state, and therefore it can be

longer than the input prediction horizon. This also promotes controller stability. [5 pp. 12-13].

The control optimization problem (4.2) resembles the bound constrained problem (3.9). The objective function (4.2a) is in quadratic form and it is subject to equality constraints (4.2b)-(4.2d) and bound constraints (4.2e)-(4.2g).

Experimental Part

The experimental part presents the practical work done in this thesis. The work includes the implementation of the new IPOPT NLP solver into the NAPCON Controller APC software and the testing conducted to validate the results and evaluate the performance of the new solver in solving the MPC optimization problem. A summary of the work is included and proposals for further study are also presented in the end.

5 Objectives

The objective of the experimental part is to interface the version 3.12 of IPOPT optimization software package with NAPCON Controller APC software, then validate its solutions and evaluate its performance against the IPOPT version 1.6 that has been utilized in NAPCON Controller before.

The implementation starts with compiling the IPOPT library with Intel Math Kernel Library (MKL) math routines and MKL PARDISO sparse linear solver. Then the necessary interface functions are updated or implemented in the NAPCON Controller to utilize the IPOPT v. 3.12. Finally, a testing program is developed to execute the controller algorithm and implement step changes to the process in repeatable and reliable fashion.

The validation is conducted by comparing the solutions of IPOPT v. 1.6 to the solutions produced by IPOPT v. 3.12. The IPOPT v. 1.6 solutions are used as the reference as the solver has been tested and used in real industrial environments for over a decade.

The evaluation consists of two different tests. In cold start test the MPC optimization task is solved once with different numbers of variables. In the warm start test the process is run in closed loop simulation and the MPC optimization task is solved continuously. The performance is evaluated by comparing the used CPU time and

iterations. All the tests will be conducted on a hydrogen treatment process model that is simulated with the NAPCON Controller software using internal Laplace models.

6 Software

The implementation of the experimental part utilizes many applications from the NAPCON Suite solution, which is an automation technology software suite for process industry developed by Neste Jacobs. The utilized applications are NAPCON Informer, NAPCON Information Manager, NAPCON Indicator and NAPCON Controller. Figure 6-1 shortly describes the functions of these applications and the data flows between them.

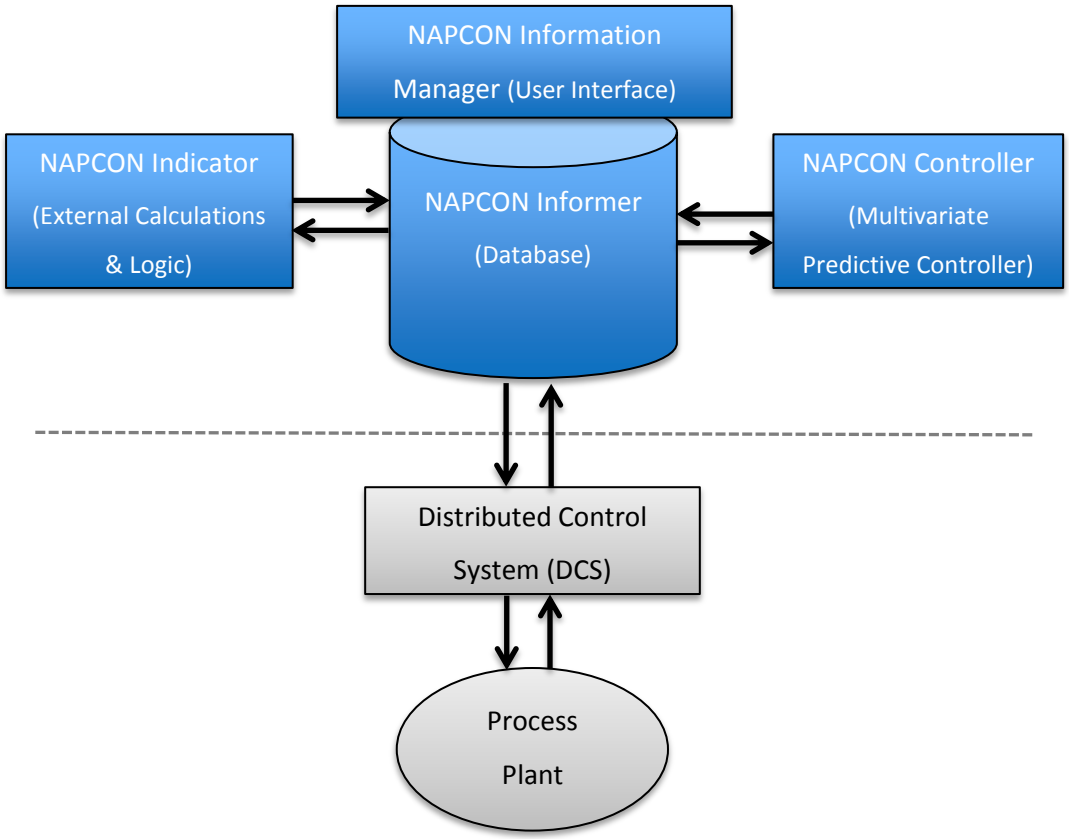

Figure 6-1. Functions and data flows of NAPCON Suite solution components.

The NAPCON Informer implements the data storing and transferring functions in the NAPCON Suite solution between different NAPCON applications but also enables communication with a DCS.

The NAPCON Information Manager acts as the user interface for the NAPCON Informer database and enables the user to examine the real-time, as well as, the historical data.

The NAPCON Indicator provides a calculation framework that can be used to implement custom calculations and logical functions to further customize the operation of NAPCON Controller.

The NAPCON Controller is a multivariate predictive process controller software that uses the receding horizon control strategy to control the process. In this thesis, the optimization of the control problem is solved using IPOPT interior point solver coupled with Intel MKL routines that implement the necessary mathematical operations.

These software applications and other utilized components are further described in the following subchapters.

6.1 NAPCON Controller

NAPCON Controller (NC) [53] is a multivariate model predictive advanced process controller software developed by Neste Jacobs and the product is sold as part of NAPCON Suite software package [54]. Majority of NC implementations are in oil refining and petrochemical industry, including a large-scale dynamic real-time optimization of an ethylene plant [55]. However, recently NC has also been used in whey powder production process [56].

NC utilizes the receding horizon control strategy and solves a MPC optimization problem similar to (4.2). In this thesis, the IPOPT interior point solver is used to solve the MPC optimization problem.

NC utilizes 5 different variable types: controlled variable (CV), manipulated variable (MV), manipulated variable constraints (MVC), disturbance variables (DV) and predicted variable (PV).

The CVs have 3 different subtypes: target CV, maximum CV and minimum CV. Target CV implements a target value for a measurement that the controller tries to reach by adjusting MVs. Maximum and minimum CVs, on the other hand, implement limits that influence the control only when the measurement gets near enough or over the limit. The controlled measurement is therefore allowed to fluctuate freely on one side of the limit and penalized on the other.

The MVs typically represent the setpoints of PID controllers implemented in the DCS. When spare degrees of freedom are available, MVs can be used to optimize the process by steering them towards an optimal value that can be calculated or given by the operator manually.

The MVCs act as additional operational limits for the MVs that allow the controller to take in consideration other constraining factors related to a specific MV such as valve openings.

The DVs implement a model between a calculated or measured disturbance and a CV so that the effects of the disturbance can be compensated in the control actions.

The PVs act as multiple input soft sensors that can be used to calculate and display key indicator values from the process or as calculated measurements for CVs and DVs.

The variable models that represent the process dynamics can be given either as Laplace models or discrete models. The model parameters can be changed online thus enabling the controller to transition into different operating points without any down time.

NC can be connected through NAPCON Informer to any automation system (DCS) that supports OPC/OPC UA communication. The NAPCON Informer acts as an interface to the real-time database that contains the all the variables and parameters related to the NC application. This enables the online tuning for controller and model parameters. The controller can also be configured using static and dynamic definition files which define the allowed values for the parameters as well as the variables and models used in the controller application. The structure of the controller solution is illustrated in Figure 6-2.

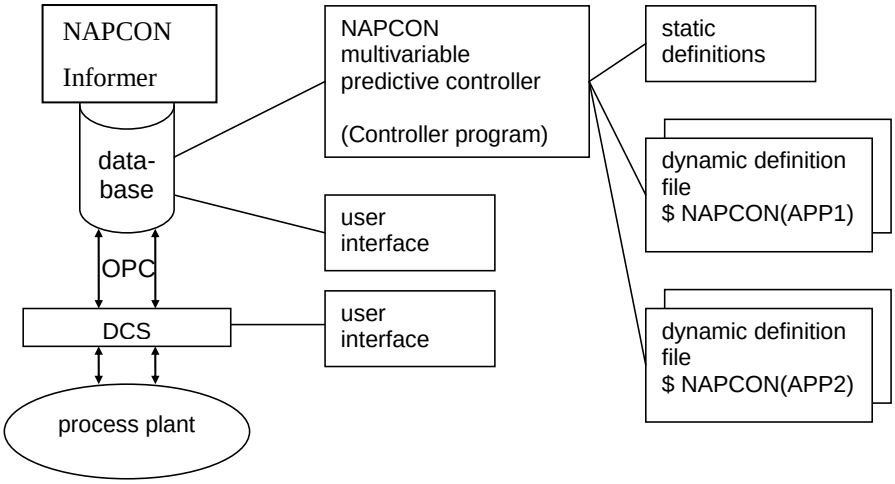

Figure 6-2. Structure of the NAPCON Controller solution.

NAPCON Controller version from NAPCON Suite 8.0 is used as the basis for the implementation in this thesis.

6.1.1 IPOPT optimization software package

IPOPT (Interior Point Optimizer) [42] is an open-source optimization software package developed at Carnegie Mellon University and it is available free of charge under Eclipse Public License. The package is intended for large-scale nonlinear optimization and it utilizes an interior point line search filter method [13] to solve optimization problems. The software is developed in C++ but includes additional interfaces for AMPL, C, FORTRAN, Java, R and MATLAB.

In addition to the state-of-the-art interior point algorithm, the software package implements special features such as derivative checker, Quasi-Newton approximation for second derivatives, warm start capabilities and sIPOPT NLP sensitivity analysis toolbox.

The software package has a large range of options the user can adjust to customize the optimization process. Notable features implemented in this package include automatic

gradient-based scaling, utilization of constant constraint Jacobians and constant Hessian, multiple barrier parameter update strategies including Mehrotra's algorithm, Fiacco-McCormick strategy and adaptive update strategies utilizing Mehrotra's probing heuristic, LOQO's centrality rule or quality-function, Quasi-Newton approximations with BFGS and SR1 methods and warm start initialization with primal and dual variables. Many other options are also available and for more information see [42].

For this thesis, IPOPT is used in NAPCON Controller to solve the MPC optimization task. The optimization algorithm is described loosely in Chapter 3.4 and in more detail in [13].

IPOPT version 3.12 is used in this thesis and it is compiled with Intel Math Kernel Library that implements mathematical operations to handle, for example, matrix calculations.

6.1.2 Intel Math Kernel Library

Intel Math Kernel Library (MKL) [57] is an optimized math library for Intel and other compatible processors. It is backwards compatible to older processor architectures by including multiple code paths. The library implements highly vectorized and threaded linear algebra, fast fourier transform, vector math and statistics functions that utilize industry-standard BLAS (Basic Linear Algebra Subroutines) and LAPACK (Linear Algebra PACKage) APIs (Application Programming Interface). Additionally, MIT's FFTW C interface for fast fourier transforms is supported.

The IPOPT optimization package is compiled with Intel MKL BLAS, Intel MKL LAPACK and Intel MKL PARDISO. However, only MKL BLAS and MKL PARDISO sparse linear solver are utilized in this thesis. The LAPACK module provides factorization and solver routines but is only used for quasi-Newton approximations in IPOPT (which are not utilized in this thesis). The BLAS module implements vector-vector, matrix-vector and matrix-matrix operations for single and double precision, real and complex types. The PARDISO solver is used in solving the linear systems produced by IPOPT.

6.2 NAPCON Indicator

NAPCON Indicator is a calculation platform developed by Neste Jacobs Oy. NAPCON Indicator consists of NAPCON Calculation Framework and NAPCON HistCalc software applications for online and offline calculations, respectively. The NAPCON Indicator utilizes Microsoft's .NET Framework thus providing language interoperability and allowing the use of external libraries in the calculations. The calculations and logical operations are implemented in a C#-file that utilizes the calculation framework interface. This file is then compiled into a DLL-file and the calculation module is installed as a Windows service (Calculation Framework) or a scheduled task (HistCalc) that is linked to the DLL-file.

Generally, the main purpose of NAPCON Indicator is to perform user-defined calculations and logical functions, such as calculating average value of multiple measurements or setting up special conditions for controller to turn off eg. key measurements or actuators become unusable. The calculation results are stored in the NAPCON Informer database, where they are available to other software such as NAPCON Controller or NAPCON Simulator. Features of the NAPCON Calculation Framework include custom scheduling of the calculations, diagnostics, automatic validity handling and analyzer signal treatment. NAPCON HistCalc is mainly used for offline calculations.

Figure 6-3 illustrates how NAPCON Indicator can be used together with NAPCON Controller. NAPCON Indicator reads data from variables in the Informer database and then performs the calculations according to the set schedule. The calculation results are then written in the database where they can be read by the NAPCON Controller and utilized when the control calculations are executed.

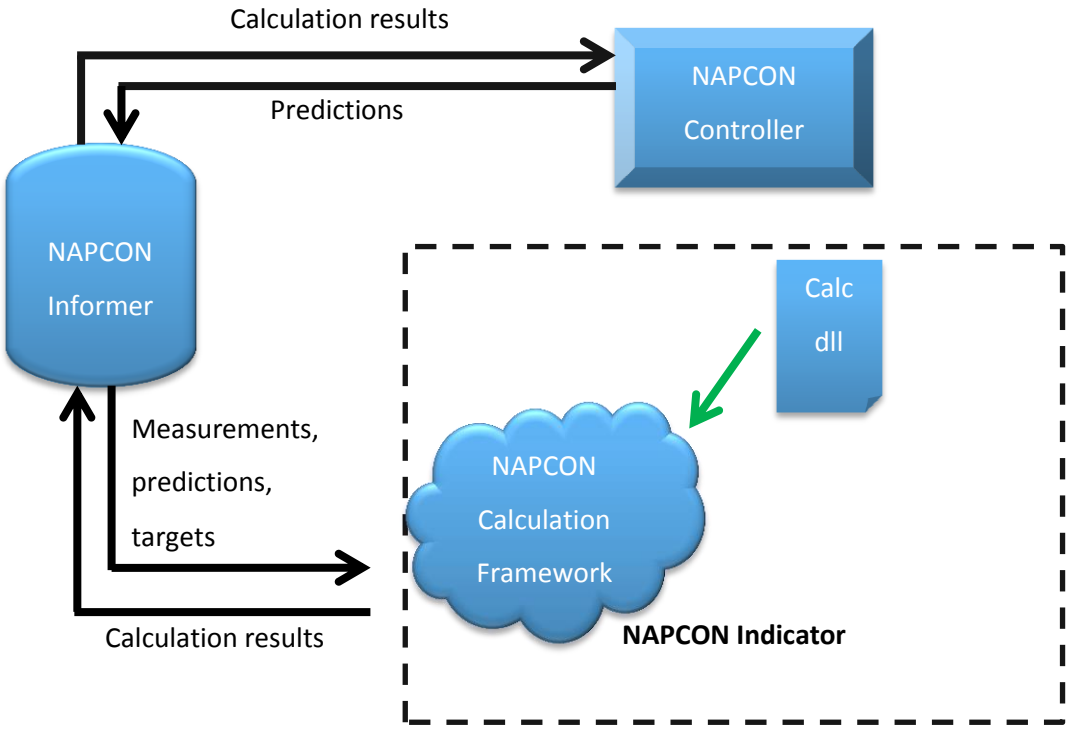

Figure 6-3. Example of a NAPCON Indicator setup. Black arrows describe data flows.

NAPCON Indicator is used in this thesis to implement the testing environment for the NAPCON Controller. The Calculation Framework is used to handle the execution of the NAPCON Controller algorithm and applying of the testing sequence of step changes. The implementation is further described in Chapter 7.

6.3 NAPCON Informer

NAPCON Informer [58] is an information management system developed by Neste Jacobs Oy. The system consists of three components: NAPCON OPC UA Server, NAPCON History Writer and NAPCON History Database. NAPCON UA Server is a real-time database with full scale interoperability from shop floor to ERP systems using OPC UA communication protocol. NAPCON History Database is a process historian for large scale data storage. NAPCON History Writer acts as an intermediary between the NAPCON OPC UA Server and NAPCON History Database. NAPCON Informer structure with data flows is presented in Figure 6-4.

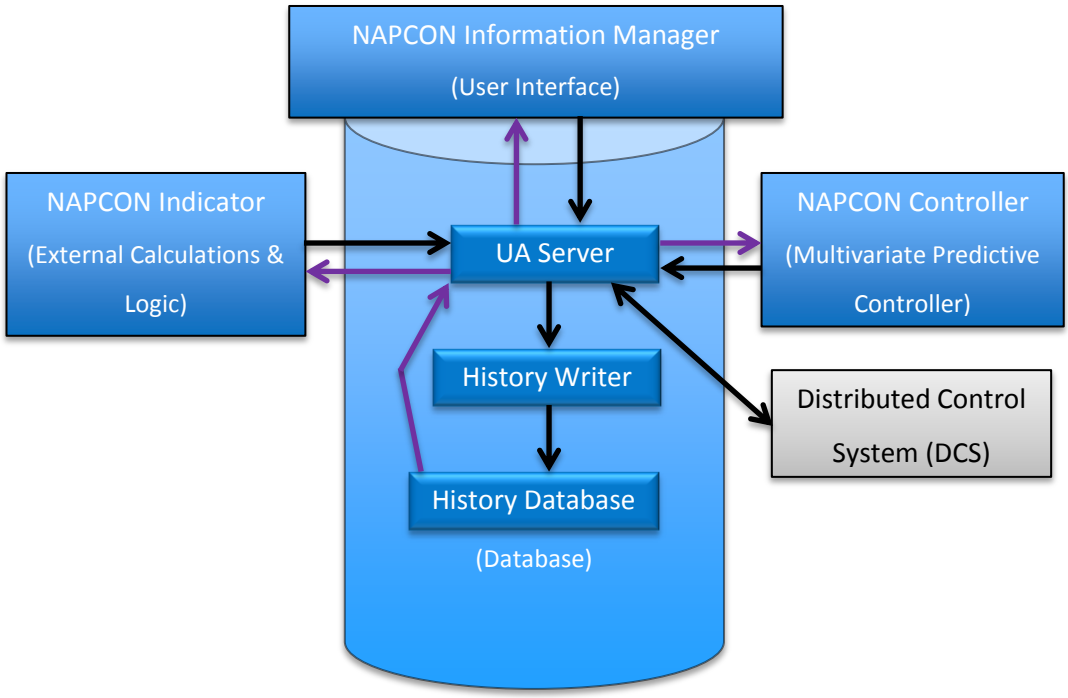

Figure 6-4. NAPCON Informer structure. Black arrows correspond to writing data flows and violet arrows to requested data flows.

NAPCON Informer enables data communications to other applications through various interfaces. Communication protocols supported are OPC UA, OPC DA, ODBC, SQL Server and Oracle. NAPCON Informer is built on Microsoft's .NET technology which provides language interoperability. NAPCON Informer supports secure communication.

NAPCON Informer acts as a data storage and intermediary between the NAPCON Controller, NAPCON Indicator and the DCS system. The fetched data from NAPCON History Database will be transferred directly to the NAPCON UA Server while the data stored into the NAPCON History Database passes through the NAPCON History Writer.

6.4 NAPCON Information Manager

NAPCON Information Manager (NIM) [59] is a software application that provides an interface to the NAPCON Informer information management system. NIM is used for monitoring, managing and maintaining the data. NIM also supports historical data which can be fetched from the NAPCON History Database and displayed in a trend view. Information Manager utilizes node lists in which the user can add any number of variables and view their real-time values. Numerical data can also be displayed in a trend view which updates in real-time and allows any number of trends at the same time. The trend view also supports viewing of historical data stored in the NAPCON History Database. The list and trend views can be saved to the hard drive for later use.

7 Implementation

The first part of the implementation was to compile the IPOPT source code (see [30] for how to obtain the code) with the Intel MKL math library. This was done in Windows environment using Microsoft Visual Studio 2012 (VS2012). The IPOPT package included example Visual Studio projects which were used as the basis for the VS2012 solution. The solution was configured to use the MKL BLAS math routines by including the MKL library (mkl_rt.lib) as additional dependency in the project linker settings. MKL PARDISO linear solver was enabled in the project by defining the "HAVE_PARDISO_MKL 1" token in the IPOPT project configuration file (config.h). If other math routine libraries, for example METIS, MUMPS or HSL, are not installed, the tokens corresponding to those libraries should be undefined in the configuration file. Additionally, if only a specific linear solver will be used, the "HAVE_LINEARSOLVERLOADER" token is recommended to be undefined to disable the dynamic solver loader.

The next part was to implement the necessary IPOPT Fortran interface functions to NC source code. There was a mistake in the original Fortran interface code of the IPOPT library that prevented public access to the interface functions when using the DLL. This problem was solved by exporting the IPOPT Fortran interface functions using "__declspec(dllexport)" -functionality of Visual C++ (for more information see [60]). Additionally, it is important to note that the Fortran interface functions in the IPOPT library assume that string length is passed after all the other arguments, which means that "NOMIXED_STR_LEN_ARG" attribute has to be used on the Fortran side (by default or by defining it individually for each function interface).

The interface provides functions for creating, solving and deleting the optimization problem, and adding options. The functions responsible for evaluating the objective function and its gradient, evaluating the constraint function and its Jacobian, and evaluating the Hessian of the Lagrangian function are defined in the program calling the IPOPT, in this case NAPCON Controller, and passed to the solver as function pointers through the IPOPT optimization problem object. IPOPT calls these evaluation functions and the mathematical routines from the MKL library when the optimization problem is being solved.

To utilize the warm start features of the IPOPT solver, the previous optimization problem solution has to be saved for the next control cycle. This has been done in NAPCON Controller before but not including the bound and constraint multipliers (primal and dual variables v^k, u^k, see Chapter 3.4 for more information). New vectors were created for these values to be stored and passed between control cycles. Only the version 3.12 of IPOPT utilizes the bound multipliers in the warm start mode.

The final part was to implement a testing platform to reliably and repeatedly execute the tests. The problem was that at the time there was no practical way to execute a sequence of step changes for the NAPCON Controller and be sure that they were executed at specific times so that the tests could be repeated. The problem was solved by implementing a single NAPCON Calculation Framework service that handles both the execution of the NAPCON Controller algorithm and the execution of the step sequence while tracking the number of executed control cycles. The sequence can be read from a file where the user needs to specify the names of the variables with the new values for the variables (steps) and the control cycle numbers when the steps are executed. Each service execution cycle the service calculations are executed, the program checks if there are any step changes to be implemented during that control cycle and implements the changes to the NAPCON Informer database if there are any. After that the controller algorithm is executed and control cycle counter is increased. This is repeated as long as the service is on. The process unit model is simulated inside the controller algorithm using internal models.

Figure 7-1 illustrates data flows in the experiment system. NAPCON Calculation Framework (NCF) is used to implement step changes to CV targets, CV limits and MV limits and then used to call the NAPCON Controller (NC) algorithm so that all the changes will happen on specific control cycles and in correct order. NCF writes the step changes into the UA Server database, where the NC can read them. When the control cycle begins NC reads all the necessary data from the UA Server and forms the MPC optimization problem. NC then calls the IPOPT to solve MPC optimization problem. IPOPT uses evaluation functions implemented in the NC and the mathematical routines in the Math Kernel Library to find the solution. NC then writes the results into the UA Server database from where the data is transferred to the History database through the

History Writer. The data is accessed through the UA Server can be viewed in the NAPCON Information Manager (NIM).

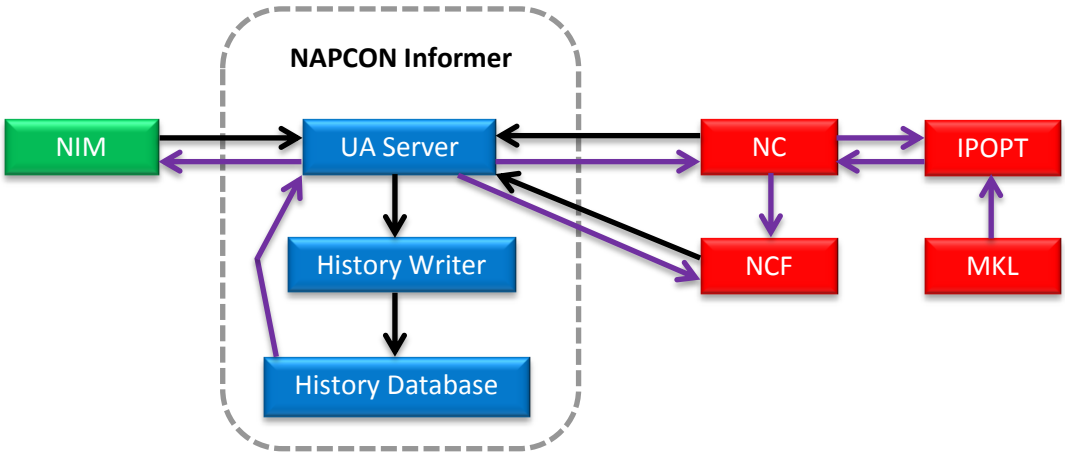

Figure 7-1. Data flows of the experiment system. Black arrows correspond to writing data flows and violet arrows to requested data flows. The software applications used in the system are NAPCON Information Manager (NIM), NAPCON Informer, NAPCON Controller (NC), NAPCON Calculation Framework (NCF), IPOPT and Math Kernel Library (MKL).

8 Testing and Evaluation

The testing and evaluation is done using two different optimization tests: the cold start test and warm start test. The goal is to assess the performance of the solvers and also ensure that the solutions are good, as in correct controller behavior is verified.

The cold start tests constitute as a worst case scenario test where the controller starts in a very unfavorable state in which many constraints are violated and the initial guess for optimization problem is bad. The test is run with different amounts of variables to also see how the problem size affects the optimization.

In the warm start test, the NAPCON Controller is used to control the test case process unit and a sequence of step and limit changes is executed to assess how the solvers perform in real time MPC optimization. Warm start optimization utilizes the solutions from the previous iteration to solve the current optimization problem faster. The test is used to see how normal control actions and difficult states affect the performance in continuous warm start optimization.

8.1 Testing criteria

In both tests the performance is evaluated by CPU time used to execute the controller algorithm (CPU time is the time used by the processor of the computer to execute the given task) and the number of iterations used to solve to optimization problem. Moreover, in the cold start tests the solver solutions are analyzed in more detail.

In the cold start optimization test the time used by the CPU is recorded for the whole controller algorithm execution but also just for the optimization task. Several key values of the optimization task are also presented: objective function value, optimality error and constraint violation. Additionally, the objective function value over the solver iterations is compared to have an illustration on how the solvers advance towards the solution. The solutions of the two different IPOPT versions are compared and the

difference is presented as average Relative Standard Deviation (RSD) of each MV step for all the MVs.

In the warm start optimization, the CPU time of the whole NAPCON Controller algorithm and the number of iterations are tracked for the whole test sequence. The performance is evaluated by comparing the CPU time and the number of iterations of both solvers. Additionally, the effects of different types of disturbances are studied.

8.2 Test case

The test case is a hydrogen treatment process unit presented by Saarela in his master's thesis [61]. The process unit is a typical petroleum refinery unit that is used in liquid fuel production. The hydrogenation process is conducted in a trickle bed reactor with 3 beds where the hydrogen gas and the liquid feed flow through a solid catalyst. The reactor bed temperatures and feed flows are controlled to achieve optimal circumstances for the hydrogenation reaction. At the bottom of the reactor the gas and liquid phases are separated. The gas phase is directed to the gas circulation system where it is treated to be used again in the reactor. The liquid phase is partly directed to the dilution circulation where it is further divided into the reactor cooling circulation and the feed dilution circulation. The cooling circulation temperature is controlled with a heat exchanger. The dilution circulation is heated with a heat exchanger and then mixed with the feed again at the top of the reactor. Rest of the liquid phase is transferred to the isomerization unit through a stripper for further processing. The process diagram of the unit is presented in Figure 8-1 with descriptions of the variables in Table 8.1.

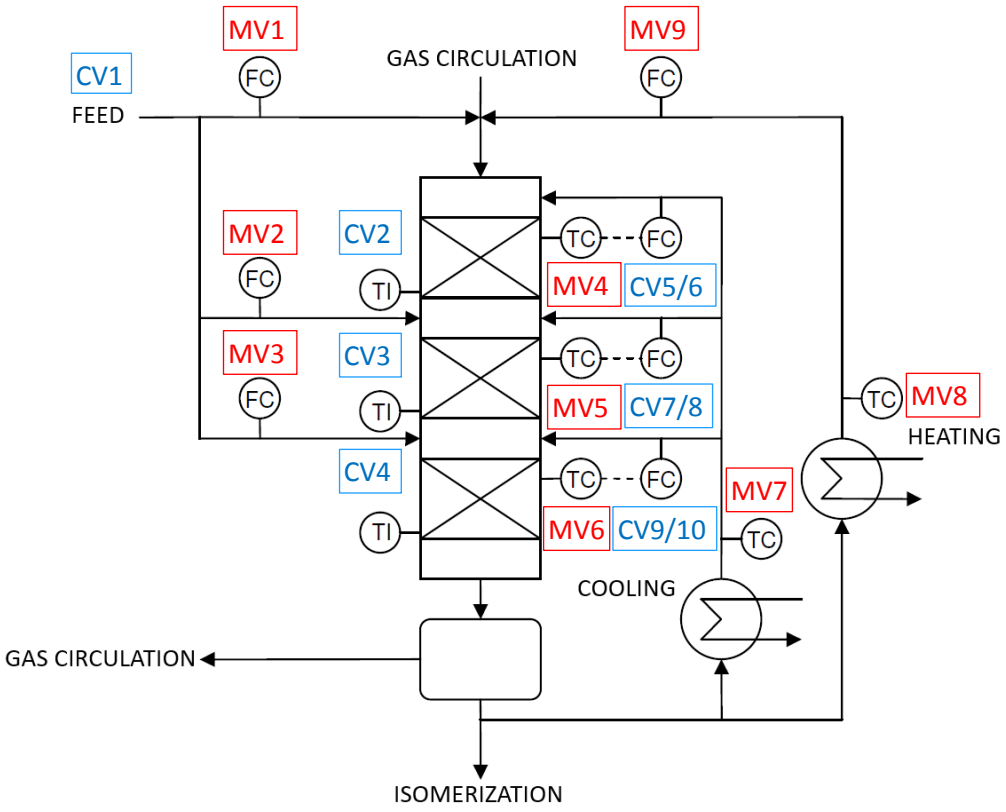

Figure 8-1. The process diagram of the test case hydrogen treatment unit.

Table 8.1. Descriptions of CVs and MVs used in the hydrogen treatment test process.

Number	Name	Description	Unit
CV1	FC001	Total feed	t/h
CV2	TI101	Bed 1 bottom temperature	°C
CV3	TI102	Bed 2 bottom temperature	°C
CV4	TI103	Bed 3 bottom temperature	°C
CV5	FC107VN	Bed 1 minimum cooling valve position	%
CV6	FC107VX	Bed 1 maximum cooling valve position	%
CV7	FC108VN	Bed 2 minimum cooling valve position	%
CV8	FC108VX	Bed 2 maximum cooling valve position	%
CV9	FC109VN	Bed 3 minimum cooling valve position	%
CV10	FC109VX	Bed 3 maximum cooling valve position	%
MV1	FC002	Bed 1 feed	t/h
MV2	FC003	Bed 2 feed	t/h
MV3	FC004	Bed 3 feed	t/h
MV4	TC104	Bed 1 top temperature setpoint	°C
MV5	TC105	Bed 2 top temperature setpoint	°C
MV6	TC106	Bed 3 top temperature setpoint	°C
MV7	TC201	Cooling circulation temperature setpoint	°C
MV8	TC202	Dilution circulation temperature setpoint	°C
MV9	FC203	Dilution circulation flow setpoint	t/h

8.2.1 Models

The hydrogen treatment process is modelled with Laplace transfer functions. The process includes 9 MVs, 10 MVCs, 4 target CVs, 6 constraint CVs and 4 DVs. However, to make the testing procedure simpler to implement, the DVs and MVCs were not utilized in the tests. The MVCs do not add complexity to the optimization task since they are processed before the task is passed to the solver. Similarly, the DVs are not very interesting in terms of the optimization task, since they do not contain any decision variables. This makes them essentially constants added to the CV values.

The Laplace functions are of the following form:

$$G(s) = \frac{G_p(T_{LD}s + 1)e^{-T_D s}}{(T_1 s + 1)(T_2 s + 1)}, \quad \textbf{(8.1)}$$

where G_p is the process gain, T_{LD} is the lead time constant, T_D is the dead time constant, T_1 is the first order model time constant and T_2 is the second order model time constant. The model parameters for the CV-MV models of the hydrogen treatment process are presented in Table 8.2, Table 8.3 and Table 8.4.

Table 8.2. Target CV Laplace model parameters from MV1-MV5 used in the hydrogen treatment test process. Time scale is seconds.

	MV1	MV2	MV3	MV4	MV5
CV1	G_p 1 T_1 210	G_p 1 T_D 30 T_1 210	G_p 1 T_D 30 T_1 360		
CV2	G_p 1.93 T_D 280 T_1 1020 T_2 100			G_p 1.0 T_D 840 T_1 2100 T_2 420	
CV3		G_p 1.84 T_D 300 T_1 1140 T_2 30		G_p 0.025 T_D 2100 T_{LD} 99999 T_1 2400 T_2 2400	G_p 1.04 T_D 480 T_1 3600 T_2 420
CV4			G_p 1.7 T_D 400 T_1 1140 T_2 30	G_p 0.012 T_D 2100 T_{LD} 99999 T_1 2400 T_2 2400	G_p 0.025 T_D 1080 T_{LD} 99999 T_1 3300 T_2 3300

Table 8.3. Target CV Laplace model parameters from MV6-MV9 used in the hydrogen treatment test process. Time scale is seconds.

	MV6	MV7	MV8	MV9
CV1				
CV2		G_p 0.065 T_D 450 T_1 1050 T_2 240	G_p 0.83 T_D 300 T_1 1200 T_2 810	G_p -0.13 T_D 2250 T_1 2400
CV3		G_p 0.095 T_D 300 T_1 900 T_2 870	G_p 0.75 T_D 960 T_1 1080 T_2 1080	G_p -0.13 T_D 2250 T_1 2400
CV4	G_p 1.04 T_D 1200 T_1 2100 T_2 420	G_p 0.125 T_D 300 T_1 1020 T_2 1020	G_p 0.68 T_D 1680 T_1 1140 T_2 1080	G_p -0.13 T_D 2250 T_1 2400

Table 8.4. Constraint CV Laplace model parameters from MV7-MV8 used in the hydrogen treatment test process. Time scale is seconds.

	MV7	MV8
CV5	G_p 0.44 T_D 150 T_1 1350 T_2 900	G_p 5.9 T_D 60 T_1 4200 T_2 1200
CV6	G_p 0.44 T_D 150 T_1 1350 T_2 900	G_p 5.9 T_D 60 T_1 4200 T_2 1200
CV7	G_p 0.3 T_D 150 T_1 1350 T_2 900	
CV8	G_p 0.3 T_D 150 T_1 1350 T_2 900	
CV9	G_p 0.3 T_D 150 T_1 1350 T_2 900	
CV10	G_p 0.3 T_D 150 T_1 1350 T_2 900	

From the model parameters presented in Table 8.2, Table 8.3 and Table 8.4 it can be noticed that the process responses are very slow as the time constants and dead times in many cases are very large. This makes the control of the process quite difficult as disturbances are hard to compensate because of the slow interactions from the MVs to the CVs. For the same reasons, conservative control strategy is preferred so that the model mismatch (in real process) doesn't lead to controller instability.

8.2.2 Control strategy

The original goal of the process control was to maximize the total feed (CV1) while keeping the bed temperatures (CV2, CV3, CV4) at certain points. For the purposes of making the tests simpler, the maximization of the total feed was turned off and the CV1 was used as normal target CV. The total feed (CV1) is controlled by the flow controllers FC002 (MV1), FC003 (MV2), FC004 (MV3). The bed feeds (MV1, MV2, MV3) also affect the bed bottom temperatures (CV2, CV3, CV4) greatly and, for the purposes of making the optimization task more difficult, they were made controlling MVs for the bed temperatures (but with minor priority) as opposed to the original control strategy where they only had a feed forward connection to the bed bottom temperatures. The bed bottom temperatures are mainly controlled by the cooling circulation that is tuned individually for each bed by temperature controllers TC104 (MV4), TC105 (MV5) and TC106 (MV6) at the top of the beds. The temperature controllers work in cascade with flow controllers FC107, FC108 and FC109. The flow controllers are included in the control strategy as valve position minimum and maximum constraints (CV5/6, CV7/8, CV9/10). The coolant temperature for the whole cooling circulation is controlled by TC201 (MV7), which affects all the bed temperatures and cooling valve position constraints. The dilution circulation temperature is controlled by TC202 (MV8) which affects the bed temperatures and bed 1 valve position constraints. The dilution flow rate is controlled by FC203 (MV9), which was originally optimized towards a certain set point, but is now set as controlling MV to increase the difficulty of the optimization. The dilution flow rate affects the bed temperatures. The control matrix of the test process is presented in Table 8.5.

Table 8.5. Control matrix for the test process. MVs can be used to control (C) or model (M) CVs, + stands for positive gain and - for negative gain.

	MV1	MV2	MV3	MV4	MV5	MV6	MV7	MV8	MV9
CV1	C+	C+	C+						
CV2	C+			C+			C+	C+	C-
CV3		C+		M+	C+		C+	C+	C-
CV4			C+	M+	M+	C+	C+	C+	C-
CV5							C+	C+	
CV6							C+	C+	
CV7							C+		
CV8							C+		
CV9							C+		
CV10							C+		

8.3 Test setup

The NAPCON Controller APC software was used to control the simulated process. Unfortunately, due to time constraints on the thesis, external simulation software could not be setup to simulate the process and therefore the simulations were carried out using the internal CV-MV Laplace models of the NAPCON Controller. This simplifies the test setup greatly but also removes the possibility to add noise or other disturbances to the measurements. As a result, the control and simulation models have no model mismatch. However, this has limited effect on solving the optimization problem since the measurements determine only the starting point for the optimization. Additionally, the variable constraints are handled as soft constraints and therefore the measurements can't cause the optimization problem to become infeasible.

The NAPCON Controller was configured so that the internal time unit for one control cycle was 60 seconds. However, since the responses in the hydrogen treatment process are very slow, the simulation was sped up to 15 times of real time speed (eg. 1 second of real time corresponds to 15 seconds of time in simulation). This was accomplished by adjusting the calculation service (NAPCON Calculation Framework module) execution loop interval. Since the internal calculations of the controller assumed that each control

cycle was 60 seconds apart, executing the algorithm more frequently than every 60 seconds speeds up the simulation (in relation to the real time).

8.3.1 Cold start test

The cold start optimization test was run 3 times for 3 different problem sizes for both of the algorithms to test the effect of problem size to the performance and to reduce uncertainties. The number of variables in the optimization problem includes CV predictions, MV steps, constraint variables and slack variables. The size of the problem was altered by decreasing the MV step stacking factor and increasing the number of unique MV steps while keeping the input horizon the same ([Unique MV steps] * [MV step stacking factor] = [Input horizon]). Table 8.6 summarizes the parameter values and the problem sizes used in the test.

Table 8.6. The cold start optimization test parameters.

Unique MV steps	MV step stacking factor	Input horizon	Number of variables
20	6	120	1140
40	3	120	2040
60	2	120	2940

The test scenario is intended to cause conflict with the control objectives so that the optimization problem becomes more difficult to solve as there are many opposing forces trying to move the MVs to different directions. The scenario is described here.

The bed bottom temperature measurements are slightly over the target values (CV2-4), which causes pressure for MV1-8 to lower their values and for MV9 to go up. Feed flow measurement is below the target value (CV1) and this creates pressure for the MV1-3 to increase in value. Bed feed flows (MV1-3) are closing to the upper limits but still have room to maneuver. Bed temperature controllers (MV4-6) are at the lower limit. Bed 1 cooling valve position (CV5) is below the lower limit which mainly pressures MV8 to

increase its value. For beds 2 and 3 the valve positions (CV8, CV10) are at the upper limit which hinders MV7 value to go up. MV optimization tries to bring the dilution circulation temperature (MV8) down and the cooling circulation temperature (MV7) up while also trying to increase the dilution circulation flow rate (MV9).

8.3.2 Warm start test

The warm start optimization test was run for approximately 25 minutes in real time (375 minutes in simulation time) during which a sequence of step and limit changes was introduced. The changes were applied on specific control cycles, which were counted starting from the moment control of the process in the NAPCON Controller was turned on. The sequence is displayed in Table 8.7.

Table 8.7. The warm start test step change sequence. The values given in the table are changes applied to the previous value in percentage, except for the blue values, which are the actual set values for the valve position limits.

Control cycle (approx. data point) Variable - Property	Initial	5 (20)	60 (240)	90 (360)	120 (480)	180 (720)	240 (960)
CV1 - TARGET		+25.00 %					
CV5 - MIN	6.00 %		10.00 %		0.00 %	6.00 %	
MV1 - MAX				-20.00 %	+87.50 %	-33.33 %	
MV2 - MAX				-20.00 %	+87.50 %	-33.33 %	
MV3 - MAX				-20.00 %	+87.50 %	-33.33 %	
CV8 - MAX	75.00 %				100.00 %	75.00 %	
MV4 - MIN					-3.23 %	+3.33 %	
MV5 - MIN					-4.15 %	+4.33 %	
MV6 - MIN					-5.06 %	+5.33 %	
MV8 - MAX							-4.86 %

8.3.3 IPOPT parameters

The optimization was run with both algorithms using 1.0E-07 error tolerance and 5000 maximum iterations. The IPOPT v. 1.6 was ran using Augmented Lagrangian line search while the IPOPT v. 3.12 used filter line search. Warm start was enabled for both versions but only the IPOPT v. 3.12 utilized also the bound multipliers of the previous iteration in addition to the solution. Constraint violation calculation was set to use infinity norm for both. The IPOPT v. 3.12 utilized adaptive barrier update strategy that minimizes a quality function [62] while the IPOPT v. 1.6 used the superlinear barrier update strategy presented in (3.21). The IPOPT v. 3.12 has the option to exploit linearity by asking the Hessian and Jacobian matrices only once, while the IPOPT v. 1.6 always asks them from the NAPCON Controller during each iteration. However, since the NAPCON Controller doesn't calculate Hessian or Jacobian matrices again but just passes them back as they were, this difference doesn't significantly affect the speed of the solvers. The IPOPT v. 3.12 uses Intel MKL math routines while the IPOPT v. 1.6 uses NAPCON math routines that are developed by Neste Jacobs Oy. Otherwise, default parameters were used in the solvers.

9 Results

9.1 Cold start test

The recorded CPU time usage results for the cold start optimization test rounds are presented in Table 9.1 and the averages are plotted in Figure 9-1. The iteration counts for the test rounds are not presented in the result table, since they did not differ. The iteration count plot can be found in Figure 9-2. Additionally, Figure 9-3 displays objective function value of the barrier problem as a function of iterations and the key values from the solver solutions are collected at Table 9.2. Finally, the differences in the optimization problem solutions are presented in Table 9.3 for all of the MVs in form of Relative Standard Deviation (RSD).

Table 9.1 presents the CPU times of all the cold start optimization test rounds with different amounts of variables. The same cold start test was executed each round and the results show variation in CPU time, as expected. The tests were done in a Windows environment, where a large number of other processes are executed continuously in a practically unpredictable manner and take up computer resources, which then affects the measured CPU time. To counter some of this uncertainty, each test was executed 3 times and the averages of the results from each of these test rounds were calculated and are used in the further discussion only.

Table 9.1. CPU time usage from the cold start optimization tests. The IPOPT rows show results for the optimization task only, while the NAPCON Controller (NC) rows present the time used by the whole NAPCON Controller algorithm execution.

CPU time	Test round 1	Test round 2	Test round 3	Average
1140 variables	**Time (s)**	**Time (s)**	**Time (s)**	**Time (s)**
IPOPT v. 3.12	1.863	2.009	1.939	1.937
NC v. 3.12	5.678	6.521	5.741	5.980
IPOPT v. 1.6	5.601	4.959	5.374	5.311
NC v. 1.6	12.964	11.934	12.761	12.553
2040 variables				
IPOPT v. 3.12	3.621	3.490	4.622	3.911
NC v. 3.12	9.875	9.734	11.934	10.514
IPOPT v. 1.6	14.720	15.217	15.617	15.185
NC v. 1.6	32.011	32.745	32.869	32.542
2940 variables				
IPOPT v. 3.12	6.446	6.328	6.810	6.528
NC v. 3.12	16.723	17.207	17.113	17.015
IPOPT v. 1.6	29.999	29.819	30.732	30.183
NC v. 1.6	58.469	57.034	58.703	58.069

Figure 9-1 presents the results of the Average column in Table 9.1. In all the cases, the IPOPT v. 3.12 outperforms the IPOPT v. 1.6 with or without the whole NAPCON Controller algorithm execution time included. The plots in the figure show approximately linear growth in CPU time as a function of number of variables, which is expected with interior point algorithms. Nevertheless, a closer inspection shows faster than linear growth in the plots of the IPOPT v. 1.6 compared to the plots of the IPOPT v. 3.12, which suggests that IPOPT v. 3.12 would outperform IPOPT v. 1.6 even more as the number of variables grows larger. Further research is required to find out if this growth is actually exponential or linear, since the data set in question is too small to draw definite conclusions. Although, even with linear increase in performance, the graphs suggest that IPOPT v. 3.12 will outperform IPOPT v. 1.6 in a greatly increasing fashion as the number of variables grows.

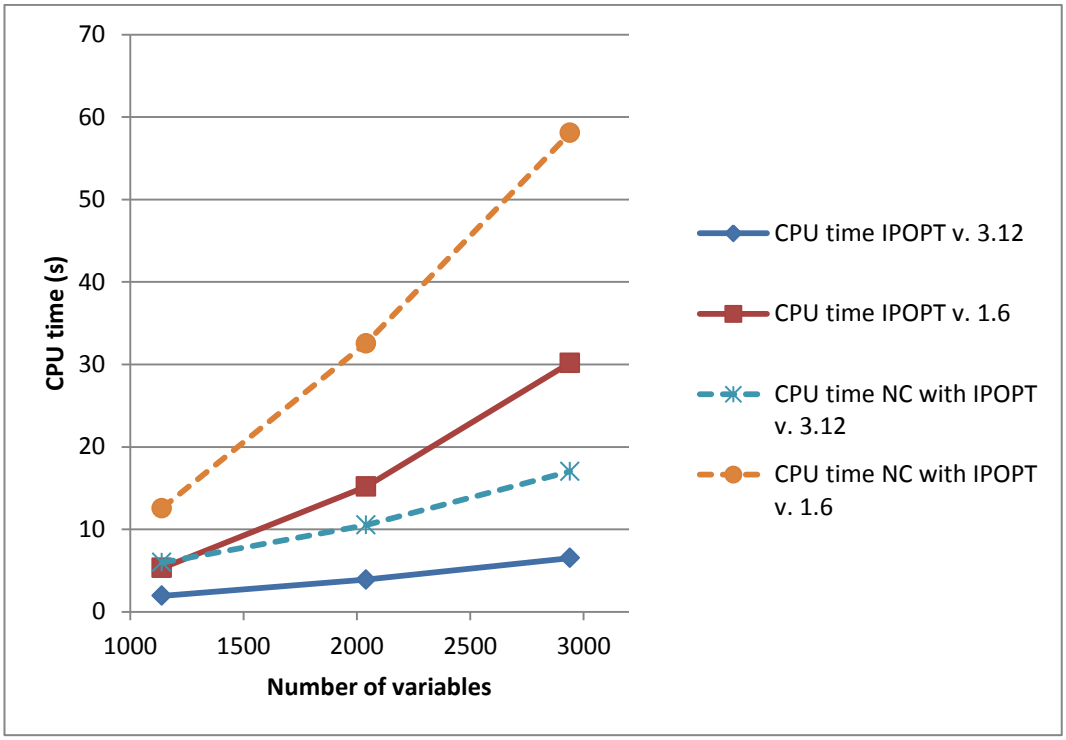

Figure 9-1. CPU time plots from the cold start optimization tests. The time is presented for the whole NAPCON Controller algorithm execution (dashed lines) and for the IPOPT solver execution (solid lines).

An interesting observation of Figure 9-1 is that even visually, the differences between the CPU times of the IPOPT v. 3.12 and v. 1.6 and the differences between the CPU times of the whole NAPCON Controller algorithm execution with those versions are not equal, which would be expected if the change in the IPOPT algorithm would be the only difference in the NAPCON Controller versions. Therefore, another element outside of the solver calculations must influence the performance of the NAPCON Controller algorithm. One possibility is the creation of the optimization problem object in IPOPT since the time recorded for the IPOPT is only measuring the time used for solving the optimization problem. Alternatively, the CPU time can be measured differently for the old and new versions. In conclusion, the cause for the difference is not clear and further study on the subject would be required to identify the cause.

Figure 9-2 shows the number of iterations against the number of variables as plots. Here the plots show a very linear curve for IPOPT v 3.12 and an approximately linear curve

with slightly slower than linear growth for IPOPT v. 1.6. For interior point algorithms the number of iterations doesn't typically change much as the number of variables changes, however, in this case the number of iterations does decrease linearly as the number of variables grows.

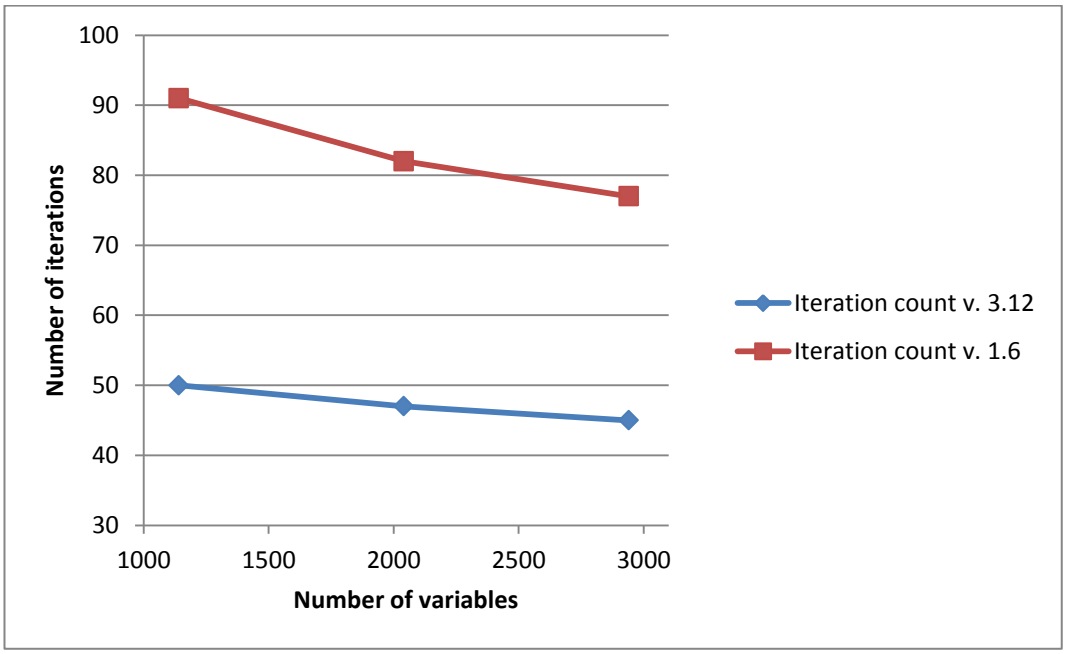

Figure 9-2. Iteration plots from the cold start optimization tests. The number of iterations used by the solvers (IPOPT v. 3.12 and v. 1.6) are presented as a function of number of variables.

In this MPC optimization case, the maximum allowed movement of MVs is a very constraining factor for the optimization since the starting points for the optimization are mostly either at the variable limits or beyond the limits. Breaking the limits induces large penalties for the objective function, which in turn causes the algorithm to try to move those variables out of the penalty zone as fast as possible by taking maximum MV steps. Therefore, the complexity of the optimization problem actually decreases as the number of variables increase, because the MV step stacking factor is decreased. The MV step stacking factor holds the values of the MVs constant for a number of control cycles determined by the factor and therefore decreasing the rate of change for the MVs. By increasing the number of unique MV steps and decreasing the MV step stacking factor,

the controller is allowed to take more steps towards the control targets when solving the optimization problem. This allows the MV trajectories to reach the steady state more easily and makes the optimization problem easier to solve. This effect can also be seen in the objective function values (presented in Table 9.2). Since the values decrease as the number of variables increases, the optimization algorithm can find a better state for the system (smaller minimum value of the objective function) in the given input interval when the MV step stacking factor decreases and the MV movement becomes less constrained.

Figure 9-3 shows objective function value of the barrier problem plotted against the number of iterations from the 1140 variables cold start test. The figure shows one way to track the progress of the optimization. Other interesting values would have been constraint violation, dual infeasibility and time used per iteration but unfortunately those values were either not available or in comparable form at the time of the tests. The figure shows that the increase in the objective function value is much steeper for the IPOPT v. 3.12 and the objective function reaches the final value much faster than for the IPOPT v. 1.6. Since the IPOPT v. 3.12 progresses much faster in the optimization, it has to take larger steps per iteration. One reason for this could be the value of the barrier parameter μ since the IPOPT v. 3.12 uses larger values for the barrier parameter than the IPOPT v. 1.6. This allows the algorithm to take larger steps, as seen from (3.18a). Further research is required to determine how much impact this actually has on the optimization performance.

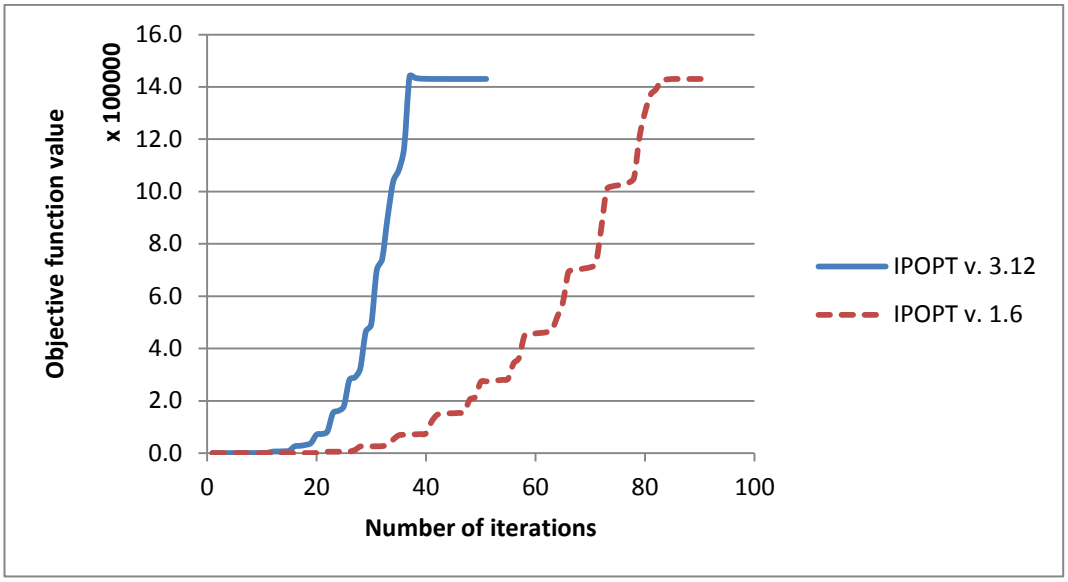

Figure 9-3. Objective function value of the barrier problem plotted against the number of iterations from the cold start 1140 variables test.

One interesting note from the figure is that even though the goal of the optimization is to minimize the objective function value, here the value increases. This happens because of the barrier term $-\mu \ln x$. The value of the term is actually negative when $x > 1$ so it contributes to the objective function by decreasing it. Therefore, when the value of x is decreased, the value of the objective function increases. On the other hand, when $x < 1$ the barrier term is positive and it increases the objective function value. Therefore the barrier parameter μ is decreased towards zero to minimize the effect of the barrier term on the objective function.

Table 9.2 presents key figures about the optimization problem. From these results it can be seen that the objective functions differ only in very minimal amounts and the constraint violation is exactly equal for both algorithms in all the cases. The optimality error, however, shows some discrepancy which is due to differences in the algorithms.

The IPOPT v. 3.12 uses internal scaling when solving the problem. The options allow the scaling to be set off but it seems that this does not work properly. When the scaling is set off, the algorithm still calculates the scaled optimality error and uses it to determine if the error is below the threshold. This usually leaves the unscaled optimality error above the threshold. Further study is required to find out if this is correct behavior of

the algorithm or a bug in the code. On the contrary, the IPOPT v. 1.6 uses only unscaled values. It should be noted that the stopping threshold has impact on the performance of the algorithms but as seen from Figure 9-3, the IPOPT v. 3.12 actually uses more iterations around the final objective function value than IPOPT v. 1.6 even though the stopping conditions for IPOPT v. 3.12 are supposed to be more relaxed.

Table 9.2. Key values from the optimization problem solutions of the cold start optimization tests.

	Objective function	Constraint violation	Optimality error (unscaled)	Optimality error (scaled)
1140 variables				
v. 3.12	1430345.21	1.4211E-14	4.9557E-06	3.114E-08
v. 1.6	1430347.31	1.4211E-14	2.8844E-08	
Relative Standard Deviation	7.3406E-05 %	0.0000 %	98.8427 %	3.8306 % *
2040 variables				
v. 3.12	717695.69	1.4211E-14	1.2144E-06	2.7746E-08
v. 1.6	717696.75	1.4211E-14	8.4050E-09	
Relative Standard Deviation	7.3950E-05 %	0.0000 %	98.6253 %	53.5005 % *
2940 variables				
v. 3.12	480122.50	1.4211E-14	1.7408E-06	8.6487E-08
v. 1.6	480123.20	1.4211E-14	5.7104E-09	
Relative Standard Deviation	7.3427E-05 %	0.0000 %	99.3461 %	87.6127 % *
* The RSD value for the scaled optimality error is calculated using the unscaled optimality error value of v. 1.6				

Table 9.3 presents the Relative Standard Deviation (RSD) of the MV steps calculated using the predicted future control moves of the NAPCON Controller with IPOPT v. 3.12 and v. 1.6. The RSD is calculated as follows

$$RSD = \frac{\sigma}{\mu}, \tag{9.1}$$

where σ is the standard deviation of the population and μ is the average of the population. The Average RSD of a MV is calculated by taking the average of the RSD values calculated for that MVs future control moves. The Maximum RSD of a MV is the maximum RSD value for that MVs future control moves.

From Table 9.3 it can be seen that the differences between the solutions produced by IPOPT v. 3.12 and IPOPT v. 1.6 are very marginal. Therefore the solutions can be held reliable and comparable. Interestingly, the average RSD does not purely increase for larger amount of variables but the second test with 2040 variables equaling to 40 unique MV steps for each MV has the largest average RSD for average of all MVs. When comparing the RSD of 2940 variable test, one reason could be that the latter half of the MV prediction horizon has already reached steady state so the solutions for that part are much closer to each other than in other parts of the horizon as the system state is then still in movement. Although for some of the MVs the average RSD is the largest in the third test with 2940 variables. The subject would require further study to be certain of the reasons.

Table 9.3. Comparison of solutions from the cold start tests. Average Relative Standard Deviation is calculated from Relative Standard Deviations of the solutions produced by IPOPT v. 1.6 and IPOPT v. 3.12 (MV steps). Maximum Relative Standard Deviation shows the largest Relative Standard Deviation of all steps for each MV.

	Average Relative Standard Deviation (%)	Maximum Relative Standard Deviation (%)
1140 variables		
MV1	1.3518E-06	2.3867E-06
MV2	8.3837E-07	1.7453E-06
MV3	1.8113E-06	4.7442E-06
MV4	9.1583E-08	1.7115E-07
MV5	9.2011E-08	1.7439E-07
MV6	1.0371E-07	1.4112E-07
MV7	1.0929E-10	3.2533E-10
MV8	1.6753E-08	3.5401E-08
MV9	2.7654E-08	4.6240E-08
Average	**4.8147E-07**	**1.0494E-06**
2040 variables		
MV1	3.2337E-04	5.5955E-04
MV2	3.2945E-04	5.6616E-04
MV3	6.3299E-04	1.1183E-03
MV4	3.3343E-07	5.9593E-07
MV5	3.7802E-07	6.4291E-07
MV6	1.0108E-06	1.0862E-06
MV7	9.1479E-07	2.4391E-06
MV8	1.1652E-07	6.9949E-07
MV9	1.0664E-06	1.8867E-05
Average	**1.4329E-04**	**2.5204E-04**
2940 variables		
MV1	7.2872E-05	1.9047E-04
MV2	7.1962E-05	1.8546E-04
MV3	1.2078E-04	4.3298E-04
MV4	3.4018E-08	1.6126E-07
MV5	5.9514E-08	1.5963E-07
MV6	3.5211E-07	3.6508E-07
MV7	6.5214E-07	1.3275E-06
MV8	1.3955E-08	8.3718E-08
MV9	1.9244E-06	9.0495E-05
Average	**2.9850E-05**	**1.0017E-04**

9.2 Warm start test

The results for the warm start optimization are presented in Figure 9-4 and Figure 9-5. The same number of data points was used to present the results for both the v. 3.12 and the v. 1.6 of IPOPT even though the actual test runs might not have been exactly equal in length since the tests were stopped manually. Due to the time constraints on the thesis, a separate control cycle (or simulation time) based data gathering system was not possible to implement and instead already existing history database system was used to gather the data. The data was collected from the NAPCON History Database using 1 second interval interpolation. This causes the data points to be in relation to real time passed (the time used by the history database), not simulation time (time used by the NAPCON Controller in simulating the process), meaning that even though all the events introduced by the test sequence happened at the same exact time in simulation, they might not have happened at the same exact time in the history data when comparing the test runs for the IPOPT versions 3.12 and 1.6. This can be seen in Figure 9-4 and Figure 9-5 as small shifts in the peaks that occurred when step changes were introduced. The reason for this difference is an oversight in designing the warm start test. The execution cycle for the NAPCON Controller algorithm was set too fast to take in account the time needed for the calculations in all cases. This results in data points shifting forward in real time in control cycles where the time for the NAPCON Controller algorithm execution takes longer than the execution cycle of the service. However, this does not affect the results in any other way than visually, since the simulations were carried out in identical fashion and enough time was given in the end of the tests to see that all the variables had reached steady state.

The results are presented in terms of iterations in Figure 9-4 and in terms of CPU time in Figure 9-5. Note, that the measured CPU time is the time recorded for executing the whole NAPCON Controller algorithm, not just the IPOPT algorithm. The number of iterations and the CPU time seem to correlate very well and similar behavior is observed when disturbances are introduced.

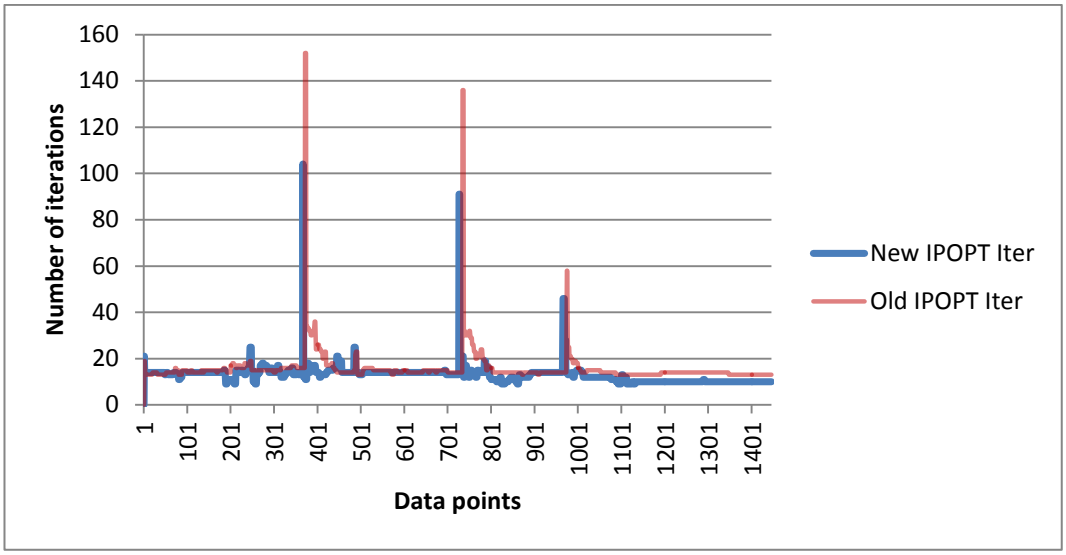

Figure 9-4. Comparison of iteration counts from the warm start optimization test. The blue plot presents iteration count for the new IPOPT (v. 3.12) and the red plot for the old IPOPT (v. 1.6).

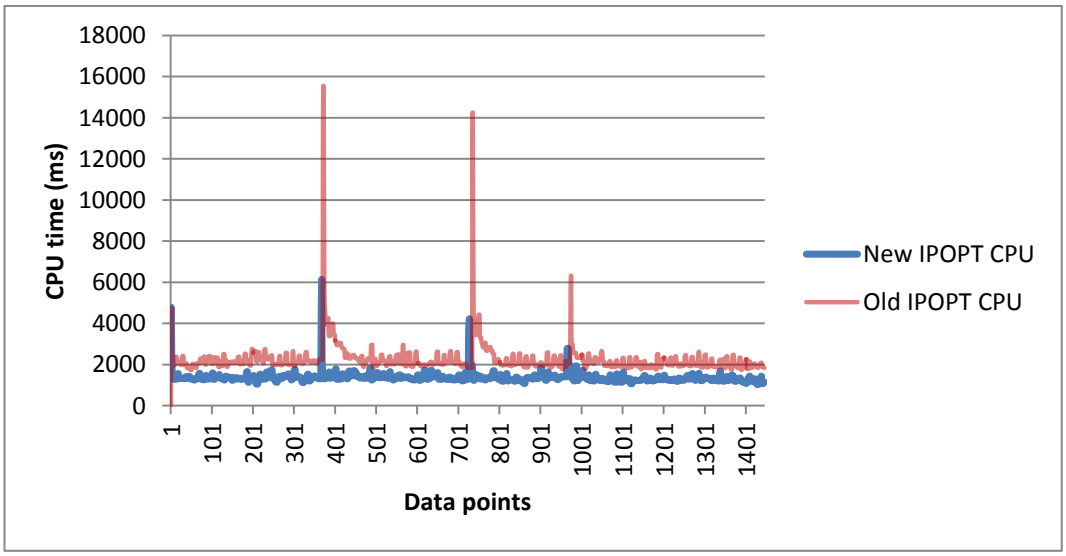

Figure 9-5. Comparison of CPU times for NC algorithm from the warm start optimization test. The blue plot presents CPU time for the new IPOPT (v. 3.12) and the red plot for the old IPOPT (v. 1.6).

The first spike around data point 1 is the start and initialization of the NAPCON Controller algorithm and both IPOPT versions use very nearly the same amount of time and iterations for this. The first step in the sequence is introduced around data point 20

when the total feed (CV1) is increased by 25% but the effect of this is unnoticeable in CPU time or iterations.

The next change to the process is introduced around data point 240 by increasing the bed 1 cooling valve position minimum constraint (CV5) from 6% to 10% making it active. This change introduced a small spike in iteration count of the IPOPT v. 3.12 but only slight variation occurred for the v. 1.6.

The first major disturbance was introduced around data point 360 when all the bed feed flow rate maximum limits were lowered by 20% making the flow rates exceed the limits. MV limits are not treated as hard constraints by the NAPCON Controller algorithm but rather as soft constraints with heavy penalty so that the optimization problem stays feasible. This heavy penalty on the limits forces the optimization to try to quickly move the MV setpoints to allowed region. This sudden large change makes the solution of the previous iteration not so good starting point since pushing the MV setpoints back to the allowed region becomes the new top priority for the optimization problem and this fights against the other goals of the optimization. The figures show that the new version performs better in both measure quantities by a large margin. Especially the CPU time shows a huge difference of almost 10000 ms (over 250%) between the IPOPT versions in favor of the v. 3.12. The CPU time and iteration count quickly settle to normal values after the disturbance for the IPOPT v. 3.12 but the IPOPT v. 1.6 comes down with a slope in both quantities. This might be indicating that the process state is in rapid movement for a while since the previous solution does not lead to a new solution as fast as in normal operation where the solution between control cycles are more similar and the warm starting works better. On the other hand, the IPOPT v. 3.12 utilizes also bound and constraint multipliers in the warm start initialization and this might help to find the new solution more quickly. However, other factors such as the barrier parameter magnitude and the update method also affect the performance and therefore further research would be required to find the exact reasons for this difference.

Next change in the system happens around data point 480, when bed 1 cooling valve position minimum constraint (CV5), bed flow rate controller maximum limits (MV1, MV2, MV3), bed 2 cooling valve position maximum constraint (CV8) and bed

temperature controller minimum limits (MV4, MV5, MV6) are relaxed. This causes only a slight spike in CPU time and iteration count for both versions.

The next major change is introduced around data point 720, when the previously relaxed constraints (CV5, MV1, MV2, MV3, CV8, MV4, MV5, MV6) are brought back to their original values. The observations from this change are similar to what occurred around data point 360.

The last step change was introduced around data point 960 when dilution flow temperature controller maximum limit was lowered by 4.86%. This caused a moderate spike in iteration count for both versions but the CPU time spike for the old version was many times larger than the minor spike that occurred for the new version. After this the process was run to the end without introducing any more disturbances.

From the test it seems that activating MV limits has a large impact on the optimization problem solving time whereas activating the CV limit didn't make such large impact when comparing the steps introduced around data points 240 and 960 where only one limit is activated. This is most likely due to the much larger penalties in the objective function that breaking the MV limits imposes compared to the CV limits. Therefore the outcome of the new optimal solution changes more when MV limits are activated and solving the problem is more difficult because the initial guess for the warm start is worse.

The warm start test shows that in continuous MPC optimization the IPOPT v. 3.12 performs much better in both terms of CPU time and in terms of iterations especially when disturbances are introduced.

10 Summary and Conclusions

This thesis has covered the basics of nonlinear optimization and MPC operation. The study conducted in this thesis focused on the performance of two interior point solvers (IPOPT v. 3.12 and IPOPT v. 1.6) in linear MPC optimization. The study covered interfacing of the IPOPT v. 3.12 interior point solver with the NAPCON Controller APC software and implementing a testing program for running the control software in such a manner that allowed repeatable and exact execution of step change sequences based on control cycles.

The test case for the MPC optimization was chosen to be a hydrogen treatment process presented by Saarela in his thesis [61]. The test configuration contained 4 target CVs, 6 constraint CVs and 9 MVs. The testing was conducted with two different tests cold start test and warm start test. The cold start test was used to test how the solvers would handle a worst case scenario where the initial starting points for the optimization were far away from the solution and the limits and targets of the controller cause conflicting goals in the optimization. The cold start test was conducted with different amounts of variables to see the effect of the number of variables in the optimization performance by changing the number of unique MV steps and MV step stacking factor so that the input horizon stayed the same. The warm start test was used to determine how the solvers performed in continuous MPC optimization while introducing disturbances.

The tests show that the IPOPT v. 3.12 outperformed the IPOPT v. 1.6 in each test by large margin. The cold start tests show that the performance difference increases as the number of variables in the optimization problem grow. The warm start tests show that the IPOPT v. 3.12 is also faster in continuous MPC optimization. When introducing disturbances IPOPT v. 3.12 solves the problem faster and returns to normal operation states quicker than IPOPT v. 1.6.

The tests performed here show that the IPOPT v. 3.12 provides a lot of improvement compared to the IPOPT v. 1.6 and therefore is a good candidate for further testing in real process environment. The testing framework developed in this thesis provides basic

functionalities for doing step testing with NAPCON Controller but still requires a lot of improvements to be used as a fully automated testing platform for NAPCON Controller.

11 Further Study

The test conducted in this thesis show clearly that the IPOPT v. 3.12 performs much better in terms of CPU time and iteration count then the older version v. 1.6 and the difference grows as the number of variables increases. This raises the question of how large application could there be for the optimization algorithm to solve it in a reasonable amount of time to be reliably used in real time process control. The usual one minute control interval used in process industry offers quite a lot of time for the calculations, and therefore another interesting research topic is new applications for MPC that might not have been possible before because of shorter control interval. Additionally, the effect of the model mismatch on the optimization performance would be important subject to study, especially in more time critical applications.

The growth in CPU time and number of iterations recorded in the tests seemed to be linear or quite close to linear. However, the sample size on this test was quite small and therefore definitive conclusions cannot be made. This leaves the door open for more research on the subject to find out if this growth is actually linear. Additionally, the number of variables was increased only by changing the MV stacking factor and therefore only the effect of increasing MV steps was studied. Consequently, the effects of increasing the number of variables in the optimization problem using more actual CVs and MVs or changing other tuning parameters could be an interesting topic for further research.

In this thesis only two very similar optimization algorithms were compared but surprisingly the difference in performance was quite large. However, these algorithms were built for nonlinear optimization and for a linear MPC only a QP problem needs to be solved. Therefore comparison to other algorithms that are more specialized for solving linear and quadratic optimization problems would be a promising topic to research if more efficiency is required.

Several issues that have come up during the tests could also demand further studying such as the NAPCON Controller execution time growth outside the solver algorithms, IPOPT v. 3.12 internal scaling issues, causes of the changes in the RSD values of the

solver solutions when using different MV stacking factors, optimization problem formulation in bound constrained form instead of equality constrained form and the impact of the barrier parameter on the solver performance.

12 Bibliography

[1] Richalet, J., et al. 1978. Model predictive heuristic control: Applications to industrial processes. Elsevier Ltd., Automatica, 14 (5), pp. 413-428. DOI: 10.1016/0005-1098(78)90001-8.

[2] Nagy, Z. K. and Allgöwer, F. 2004. Nonlinear Model Predictive Control: From Chemical Industry to Microelectronics. IEEE, 43rd IEEE Conference on Decision and Control. Vol. 4, pp. 4249-4254. DOI: 10.1109/CDC.2004.1429419.

[3] Bazaraa, M.S., Sherali, H.D. and Shetty, C.M. 2006. Nonlinear Programming: Theory and Algorithms. John Wiley and Sons p. 853. DOI: 10.1002/0471787779.

[4] Biegler, L.T. 2014. Recent advances in chemical process optimization. Wiley-VCH Verlag, Chemie-Ingenieur-Technik, 86 (7), pp. 943-952. ISSN: 0009286X. DOI: 10.1002/cite.201400033.

[5] Biegler, L. T. 2010. Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes. Pittsburgh: Society for Industrial and Applied Mathematics p. 399.

[6] Alexandrov, Oleg. Wikipedia. [Online] [Cited: Jan 15, 2015.] http://en.wikipedia.org/wiki/Convex_set#mediaviewer/File:Convex_polygon_illustration1.png.

[7] Alexandrov, Oleg. Wikipedia. [Online] [Cited: Jan 23, 2015.] http://en.wikipedia.org/wiki/Convex_set#mediaviewer/File:Convex_polygon_illustration2.png.

[8] Kuhn, Harold and Tucker, Albert. 1951. Nonlinear programming. University of California Press, Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Propability. pp. 481-492.

[9] Kuhn, Harold. 1976. Nonlinear programming: A historical view. American Mathematical Society, Nonlinear Programming, SIAM-AMS Proceedings, 9 (1), pp. 1-26. ISBN-10: 0-8218-1329-3.

[10] Nocedal, Jorge and Wright, Stephen J. 2006. Numerical Optimization. Springer New York p. 664. ISBN: 978-0-387-30303-1. DOI: 10.1007/978-0-387-40065-5.

[11] Dantzig, George B. and Thapa, Mukund N. 2003. Linear Programming: 2: Theory and Extensions. Springer-Verlag p. 448. ISBN: 978-0-387-21569-3.

[12] Wright, Margaret H. 2004. The interior-point revolution in optimization: History, recent developments and lasting consequences. American Mathematical Society, Bulletin of the American Mathematical Society, 42 (1), pp. 39-56. DOI: 10.1090/S0273-0979-04-01040-7.

[13] Wächter, A. and Biegler, L. T. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Springer-Verlag, Mathematical Programming, 106 (1), pp. 25-57. ISBN: 0025-5610. DOI: 10.1007/s10107-004-0559-y.

[14] Forsgren, A., Gill, P. E. and Wright, M. H. 2002. Interior Methods for Nonlinear Optimization. SIAM, SIAM Review, 44 (4), pp. 525-597. DOI: 10.1137/S0036144502414942.

[15] Wächter, A. and Biegler, L. T. 2005. Line Search Filter Methods for Nonlinear Programming: Motivation and Global Convergence. SIAM, SIAM Journal on Optimization, 16 (1), pp. 1-31. DOI: 10.1137/S1052623403426556.

[16] Wisconsin Institutes for Discovery. NEOS Server for Optimization. [Online] [Cited: May 29, 2015.] http://www.neos-server.org/neos/.

[17] Mittelmann, H. D. Decision Tree for Optimization Software. [Online] [Cited: May 15, 2015.] http://plato.asu.edu/guide.html.

[18] CCM, University of California, San Diago. Optimization Software. [Online] [Cited: May 15, 2015.] https://ccom.ucsd.edu/~optimizers/software.html.

[19] Gill, P. E., Murray, W. and Saunders, M. A. 2005. SNOPT: An SQP Algorithm for Large-scale Constrained Optimization. SIAM, SIAM Review, 47 (1), pp. 99-131. DOI: 10.1137/S0036144504446096.

[20] Stanford Business Software, Inc. SNOPT 6.0. [Online] [Cited: Jun 11, 2015.] http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm.

[21] Bremer, I., Henrion, R. and Möller, A. 2015. Probabilistic constraints via SQP solver: application to a renewable energy management problem. Springer Berlin Heidelberg, Computational Management Science, 12 (3), pp. 435-459. DOI: 10.1007/s10287-015-0228-z.

[22] Xie, Y., et al. 2014. Descent trajectory optimization for stratospheric airships with thermal effects. IEEE, Guidance, Navigation and Control Conference (CGNCC), 2014 IEEE Chinese. pp. 612-617. DOI: 10.1109/CGNCC.2014.7007287.

[23] Xue, N., et al. 2014. Design of a lithium-ion battery pack for PHEV using a hybrid optimization method. Elsevier, Applied Energy, 115 (1), pp. 591-602. DOI: doi:10.1016/j.apenergy.2013.10.044.

[24] The Boeing Company. . SOCS User's Guide Release 7.1.

[25] Betts, J. T. and Frank, P. D. 1994. A sparse nonlinear optimization algorithm. Kluwer Academic Publishers-Plenum Publishers, Journal of Optimization Theory and Applications, 82 (3), pp. 519-541. ISSN: 1573-2878. DOI: 10.1007/BF02192216.

[26] Mathematical Research Institute Oberwolfach; FIZ Karlsruhe. SOCS - Mathematical software - swMath. [Online] [Cited: Jun 11, 2015.] http://www.swmath.org/software/7737.

[27] Tsiotras, P. and Diaz, R. S. 2014. Real-Time Near-Optimal Feedback Control of Aggressive Vehicle Maneuvers. Springer Verlag, Lecture Notes in Control and Information Sciences. Vol. 455, pp. 109-129. DOI: 10.1007/978-3-319-05371-4_7.

[28] Patterson, M. A. and Rao, A. V. 2014. GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming. ACM, ACM Transactions on Mathematical Software, 41 (1). DOI: 10.1145/2558904.

[29] Zhao, Y. and Tsiotras, P. 2011. Density Functions for Mesh Refinement in Numerical Optimal Control. American Institute of Aeronautics and Astronautics, Journal of Guidance, Control and Dynamics, 34 (1), pp. 271-277. DOI: 10.2514/1.45852.

[30] COIN-OR. Ipopt. [Online] [Cited: May 15, 2015.] https://projects.coin-or.org/Ipopt.

[31] COIN-OR. SuccessStories - Ipopt. [Online] [Cited: Jun 11, 2015.] https://projects.coin-or.org/Ipopt/wiki/SuccessStories.

[32] Jiang, A., et al. 2015. Optimal operations for large-scale seawater reverse osmosis networks. Elsevier, Journal of Membrane Science, 476 (1), pp. 508-524. DOI: 10.1016/j.memsci.2014.12.005.

[33] Ramos, M. A., Gómez, J. M. and Reneaume, J.-M. 2014. Simultaneous Optimal Design and Control of an Extractive Distillation System for the Production of Fuel Grade Ethanol Using a Mathematical Program with Complementarity Constraints. ACS Publications, Ind. Eng. Chem. Res., 53 (2), pp. 752-764. DOI: 10.1021/ie402232w.

[34] Holmqvist, A., et al. 2014. Dynamic parameter estimation of atomic layer deposition kinetics applied to in situ quartz crystal microbalance diagnostics. Elsevier, Chemical Engineering Science, 111 (), pp. 15-33. DOI: 10.1016/j.ces.2014.02.005.

[35] Ziena Optimization LLC. Ziena Optimization LLC | Knitro (R) for solving nonlinear optimization problems. [Online] [Cited: May 15, 2015.] http://www.ziena.com/knitro.htm.

[36] Byrd, R. H., Nocedal, J. and Waltz, R. A. 2006. KNITRO: An Integrated Package for Nonlinear Optimization. Springer US, Nonconvex Optimization and Its Applications, 83 (1), pp. 35-59. ISSN: 978-0-387-30065-8. DOI: 10.1007/0-387-30065-1_4.

[37] Artelys. KNITRO - Artelys | Optimization Solutions. [Online] [Cited: Jun 11, 2015.] http://www.artelys.com/en/optimization-tools/knitro.

[38] Chen, J., et al. 2014. A Knitro-based real-time locomotion method for imitating the caterpillar-like climbing strategy. IEEE, 11th IEEE International Conference on Control and Automation, ICCA. pp. 145-150. DOI: 10.1109/ICCA.2014.6870911.

[39] Ruiz, M., et al. 2014. A progressive method to solve large-scale AC optimal power flow with discrete variables and control of the feasibility. IEEE, Power Systems Computation Conference, PSCC. pp. 1-7. DOI: 10.1109/PSCC.2014.7038395.

[40] Nahayo, F., Khardi, S. and Haddou, M. 2012. Two-aircraft dynamic system on approach. Flight path and noise optimization. Applied Mathematical Sciences, Applied Mathematical Sciences, 6 (77-80), pp. 3861-3880. ISSN: 1312885X.

[41] Hicks, G. A. and Ray, W. H. 1971. Approximation Methods for Optimal Control Synthesis. Canadian Society for Chemical Engineering, The Canadian Journal of Chemical Engineering, 49 (4), pp. 522-528. DOI: 10.1002/cjce.5450490416.

[42] Kawajir, Yoshiaki, et al. 2015. Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT.

[43] Gould, N., Orban, D. and Toint, P. 2005. Numerical methods for large-scale nonlinear optimization. Cambridge University Press, Acta Numerica, 14 (1), pp. 299-361. DOI: 10.1017/S0962492904000248.

[44] Morari, M. and Lee, J. H. 1999. Model predictive control: past, present and future. Elsevier, Computers and Chemical Engineering, 23 (4-5), pp. 667-682. DOI: 10.1016/S0098-1354(98)00301-9.

[45] Qin, S. J. and Badgwell, T. A. 2003. A survey of industrial model predictive control technology. Elsevier, Control Engineering Practice, 11 (7), pp. 733-764. DOI: 10.1016/S0967-0661(02)00186-7.

[46] Bauer, M. and Craig, I. K. 2008. Economic assessment of advanced process control - A survey and framework. Journal of Process Control, Journal of Process Control, 18 (1), pp. 2-18. DOI: 10.1016/j.jprocont.2007.05.007.

[47] Canney, W. M. 2005. Are you getting the full benefit from your advanced process control system? Hydrocarbon Processing, 84 (6), pp. 55-58. ISSN: 00188190.

[48] Allgöwer, F., Findeisen, R. and Nagy, Z. K. 2004. Nonlinear Model Predictive Control: From Theory to Application. Journal of the Chinese Institute of Chemical Engineers, J. Chin. Inst. Chem. Engrs., 35 (3), pp. 299-315.

[49] Grüne, L. and Pannek, J. 2011. Nonlinear Model Predictive Control: Theory and Algorithms. Springer-Verlag London p. 360. DOI: 10.1007/978-0-85729-501-9.

[50] Norquay, S. J., Palazoglu, A. and Romagnoli, J. A. 1998. Model predictive control based on Wiener models. Elsevier, Chemical Engineering Science, 53 (1), pp. 75-84. DOI: 10.1016/S0009-2509(97)00195-4.

[51] Cervantes, A. L., Agamennoni, O. E. and Figueroa, J. L. 2003. A nonlinear model predictive control system based on Wiener piecewise linear models. Elsevier, Journal of Process Control, 13 (7), pp. 655-666. DOI: 10.1016/S0959-1524(02)00121-X.

[52] Lawrynczuk, M. 2013. Practical nonlinear predictive control algorithms for neural Wiener models. Elsevier, Journal of Process Control, 23 (5), pp. 696-714. DOI: 10.1016/j.jprocont.2013.02.004.

[53] Neste Jacobs Oy. 2015. NAPCON Controller 8.0 Reference Manual. Porvoo:

[54] Neste Jacobs Oy. Napcon - Home. [Online] [Cited: Jan 27, 2015.] http://www.napconsuite.com/index.php?lang=en.

[55] Vetteranta, J. and et. al. 2006. Dynamic real-time optimization increases ethylene plant profits. Gulf Publishing Company, Hydrocarbon Prosessing, 85 (10), pp. 59-59.

[56] Neste Jacobs Oy. NAPCON Controller maximizes whey powder production. [Online] 2015. [Cited: Jan 27, 2015.]

http://www.napconsuite.com/data/docs/nj_valio_napcon_controller_whey_powder.pdf.

[57] Intel Corporation. Details about Intel(R) Math Kernel Library | Intel(R) Developer Zone. [Online] [Cited: Jun 2, 2015.] https://software.intel.com/en-us/intel-mkl/details.

[58] Neste Jacobs Oy. 2014. NAPCON Informer 8.0 User Manual. Porvoo: Neste Jacobs Oy

[59] Neste Jacobs Oy. 2014. NAPCON Informer 8.0 Information Manager Manual. Porvoo: Neste Jacobs Oy

[60] Microsoft. Exporting from a DLL Using __declspec(dllexport). [Online] [Cited: 04 12, 2016.] https://msdn.microsoft.com/en-us/library/a90k134d.aspx.

[61] Saarela, V. 2009. Advanced control structure for production intensification and constraint analysis in hydrotreating process. Oulu: The University of Oulu In Finnish.

[62] Nocedal, J., Wächter, A. and Waltz, R. A. 2009. Adaptive barrier update strategies for nonlinear interior methods. SIAM, SIAM Journal on Optimization, 19 (4), pp. 1674-1693. DOI: 10.1137/060649513.